UNIVERSIDAD TECNOLÓGICA ISRAEL

TRABAJO DE TITULACIÓN EN OPCIÓN AL GRADO DE:

INGENIERO EN SISTEMAS INFORMÁTICOS

TEMA: DESARROLLO DE UN SISTEMA INFORMÁTICO DE ESCRITORIO PARA LA GESTIÓN DEL INVENTARIO PARA LA EMPRESA CAR POINT.

AUTOR: EDISON MARTIN MOLINA SORIA

TUTOR: Mg. CHRISTIAN PATRICIO VACA BENALCAZAR

QUITO- ECUADOR

AÑO: 2018

DECLARACIÓN DE AUTORÍA

El documento de tesis con título: "DESARROLLO DE UN SISTEMA INFORMÁTICO DE ESCRITORIO PARA LA GESTIÓN DEL INVENTARIO PARA LA EMPRESA CAR POINT.", ha sido desarrollado por el señor EDISON MARTIN MOLINA SORIA con C.C. No. 1718161068 persona que posee los derechos de autoría y responsabilidad, restringiéndose la copia o utilización de la información de esta tesis sin previa autorización.

_____

Edison Martin Molina Soria

UNIVERSIDAD TECNOLÓGICA ISRAEL

APROBACIÓN DEL TUTOR

En mi calidad de Tutor del Trabajo de Titulación certifico:

Que el trabajo de titulación **"DESARROLLO DE UN SISTEMA INFORMÁTICO DE ESCRITORIO PARA LA GESTIÓN DEL INVENTARIO PARA LA EMPRESA CAR POINT."**, presentado por EDISON MARTIN MOLINA SORIA, estudiante de la Carrera Ingeniería en Sistemas Informáticos, reúne los requisitos y méritos suficientes para ser sometido a la evaluación del Tribunal de Grado, que se designe, para su correspondiente estudio y calificación.

Quito D. M., 16-08-2018

TUTOR

Mg. Christian Vaca

DEDICATORIA

Dedico este trabajo a mis padres Martin Molina y Patricia Soria por ser un gran ejemplo a seguir el cual me motiva a no desfallecer en todo el transcurso de mi carrera universitaria.

A mi querida esposa Andrea Haro y a mi hermosa hija Emily Molina por ser el motor que alimenta mi fuerza de voluntad y motivación para culminar con éxitos mi proyecto de tesis.

A mi tutor Ing. Christian Vaca Msc, por el apoyo incondicional como docente y como guía para la elaboración del presente proyecto de tesis.

TABLA DE CONTENIDO

DECLARACIÓN DE AUTORÍA I

APROBACIÓN DEL TUTOR II

DEDICATORIA III

TABLA DE CONTENIDO IV

LISTA DE FIGURAS VII

LISTA DE TABLAS VIII

RESUMEN X

INTRODUCCIÓN XII

Antecedentes de la situación objeto de estudio XII

Planteamiento del problema XII

Formulación del problema XIII

Justificación XIII

Objetivo General XIV

Objetivo específicos XIV

1 CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA 15

1.1 Taller Automotriz 15

1.2 Bodega 15

1.3 Inventario 16

1.4 Control de existencias 16

1.5 Kárdex 16

1.6 Sistemas de Inventario 17

1.7 Valorización de existencias del inventario 17
1.8 Métodos de valorización de existencias 17
1.9 Norma internacional de contabilidad 2 - NIC 2 18
1.10 Metodologías Agiles 18
1.11 Tipos de Metodologías Ágiles 19
1.12 Metodología Extreme Programming 20
1.13 Ciclo de visa de software en XP 20
1.14 Scrum 23
1.15 Roles 25
1.16 Herramientas de Licenciamiento Open Source 26
1.17 Java 26
1.18 Netbeans 27
1.19 PostgreSQL 28
1.20 Arquitectura MVC 28
2 CAPÍTULO II. PROPUESTA 30
2.1 Recopilación de información 30
2.2 Diagramas de Procesos 32
2.3 Especificación de Requerimientos 35
3 CAPÍTULO III. IMPLEMENTACIÓN 43
3.1 Diseño general 43
3.2 Esquema de la Base de Datos 50
3.3 Diagrama de la arquitectura del sistema 52
3.4 Diseño de interfaces 52

3.5 Estándares de programación utilizados 55
3.6 Implementación 57
4 CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES 60
4.1 Conclusiones 60
4.2 Recomendaciones 60
5 BIBLIOGRAFÍA 61

LISTA DE FIGURAS

Imagen 1. Formula del Precio Medio Ponderado 18
Imagen 2 Metodologías Ágiles 19
Imagen 3 Metodologías de desarrollo de software 20
Imagen 4 Ciclo de vida de Scrum 24
Imagen 5 Procesos y Roles de Scrum 25
Imagen 6 Resultado de Encuesta Realizada 30
Imagen 7 Diagrama de Proceso CarPoint 32
Imagen 8 Proceso de almacenamiento de repuestos 33
Imagen 9 Proceso de Almacenamiento Propuesto 34
Imagen 10 Modelo Conceptual 50
Imagen 11 Modelo Físico 51
Imagen 12 Arquitectura MVC 52
Imagen 13 Venta De Ingreso 52
Imagen 14 Ventana Principal 53
Imagen 15 Ventana Movimiento 53
Imagen 16 Ventana Maestro 54
Imagen 17 Ventana Administración 54
Imagen 18 Ventana Venta 55
Imagen 19 Estándares de programación 56

LISTA DE TABLAS

Tabla 1 Tipos Sistemas de Inventario 17
Tabla 2 Valorización de existencias 18
Tabla 3 Resumen de Historia de Usuarios 36
Tabla 4 Historia de usuarios HU1 36
Tabla 5 Historia de usuario HU2 37
Tabla 6 Historia de usuario HU3 37
Tabla 7 Historia de usuario HU4 37
Tabla 8 Historia de usuario HU5 38
Tabla 9 Historia de usuario HU6 38
Tabla 10 Historia de usuario HU7 38
Tabla 11 Historia de usuario HU8 39
Tabla 12 Historia de usuario HU9 39
Tabla 13 Historia de usuario HU10 39
Tabla 14 Tarjeta CRC Usuario 43
Tabla 15 Tarjeta CRC Perfiles 43
Tabla 16 Tarjeta CRC Clientes 44
Tabla 17 Tarjeta CRC Proveedor 44
Tabla 18 Tarjeta CRC Artículo 44
Tabla 19 Tarjeta CRC Venta 45
Tabla 20 Tarjeta CRC Compra 45
Tabla 21 Tarjeta CRC Reporte 45
Tabla 22 Tarjeta CRC Devoluciones 45
Tabla 23 Tarjeta CRC Stock 46
Tabla 24 Prueba de aceptación PA1 46
Tabla 25 Prueba de aceptación PA2 47
Tabla 26 Prueba de aceptación PA3 47
Tabla 27 Prueba de aceptación PA4 48
Tabla 28 Prueba de aceptación PA5 48
Tabla 29 Prueba de aceptación PA6 49
Tabla 30 Plan de implementación 57

Tabla 31 Requerimientos de HW/SW 58
Tabla 32 Plan de Capacitación.....59

RESUMEN

El siguiente proyecto de tesis está enfocado en el desarrollo del sistema de escritorio para la gestión del inventario de la empresa “CarPoint” con el objetivo de administrar correctamente la Bodega, ya que es manejada de forma manual originando una evidente pérdida económica.

El software permitirá un adecuado control del inventario de repuestos como también de la maquinaria nueva que ingresa al taller automotriz; además el mismo sustituirá el incorrecto e improvisado manejo manual por parte del encargado de Bodega.

Por ende, el resultado de este sistema informático que se va a implementar, brindará un manejo y control oportuno por parte de los administradores y Gerentes de la empresa “CarPoint”; mediante la generación de reportes que conllevará a un registro verídico, para evidenciar el completo stock de Bodega.

Finalmente, con la innovación del sistema informático, se obtendrá una información completamente confiable y certera, con la finalidad de cambiar el proceso del manejo de Bodega que actualmente es llevada en la empresa “CarPoint”, logrará determinar sus productos automotrices de una manera más organizada. La contabilidad manual no es adecuada y tiende a ser subjetiva; lo que ocasiona poco control de sus activos fijos; originando datos falsos e imprecisos; y provocando una quiebra posterior de la Empresa. Es por todo lo expuesto, que la Empresa “CarPoint” debe conocer en forma precisa con qué cuenta exactamente en la Bodega de su compañía para el buen desenvolvimiento de su actividad productiva.

PALABRAS CLAVES:

Bodega, Improvisado, Confiable, Imprecisos, Productiva.

ABSTRACT

The following thesis project is focused on the development of the desktop system for the management of the stocktaking of the company "CarPoint" with the aim of properly managing the warehouse, since it is handled manually, causing an evident economic loss.

The software will allow an adequate control of the stocktaking of spare parts as well as the new machinery that enters the automotive workshop; In addition, it will replace the incorrect and improvised manual handling by the manager of Bodega.

Therefore, the result of this computer system that will be implemented, will provide timely management and control by the administrators and managers of the company "CarPoint"; through the generation of reports that will lead to a true record, to show the complete stock of Warehouse.

Finally, with the innovation of the computer system, a completely reliable and accurate information will be obtained, in order to change the process of warehouse management that is currently carried out in the company "CarPoint", will be able to determine its automotive products in a more organized way . Manual accounting is not adequate and tends to be subjective; which causes little control of their fixed assets; originating false and inaccurate data; and causing a subsequent bankruptcy of the Company. It is for all the above, that the company "CarPoint" must know precisely what account exactly in the warehouse of your company for the proper development of their productive activity.

KEYWORDS:

Warehouse, Improvised, Reliable, Vague, Productive.

INTRODUCCIÓN

Antecedentes de la situación objeto de estudio

La empresa "CarPoint" nace como inspiración de un grupo de personas dedicadas al campo automotriz, para plasmar en un local el emprendimiento en este tipo de negocio, dedicado a todo tipo de marca comercial de vehículos existentes en el mercado nacional e internacional.

Se funda en Abril del 2017 en la parroquia de Tumbaco con un total de 11 socios; la empresa cuenta con un Gerente, una Asistente de Gerencia, un Jefe de Taller Automotriz, 4 ayudantes, una Secretaria y una persona encargada de Bodega.

"CarPoint" cuenta con diversos equipos para el correcto mantenimiento automotriz; entre ellos se puede nombrar 4 elevadores, equipo de diagnóstico automotriz computarizado similares a los que se ocupan en la "Corpaire" – Quito; en resumen es una inversión de un total de 1 millón de dólares en infraestructura como maquinaria automotriz.

Planteamiento del problema

"CarPoint" es una empresa nueva en el campo del Taller Automotriz, y gracias a la gran acogida que está teniendo en el mercado surge la problemática de cómo se debe llevar un manejo adecuado de su Bodega automotriz.

En un Taller Automotriz moderno, la Bodega es administrada con un sistema informático el cual indica que repuestos están por terminarse, como también que productos no están saliendo; cabe mencionar que una empresa de esta magnitud, tiene validación para el ingreso del software con su clave y contraseña para el acceso. A su vez la Gerencia con la adecuación de este proceso puede reconocer el margen de ganancia que tiene con cada tipo de repuesto, como también quien lo distribuye. En resumen tienen el control total y sistematizado de forma rápida y precisa de su Bodega.

En la actualidad existe un caos en la coordinación del manejo del inventario, donde la cantidad de repuestos que ingresan es cada vez más grande y está aumentando en forma acelerada, por tal motivo existe una descoordinación total al momento de generar un código por cada producto nuevo.

El personal de Bodega constantemente está actualizando la lista de códigos porque un repuesto puede tener uno o más códigos, existiendo mucha duplicidad de información.

Adicionalmente, por la ausencia de procedimientos eficientes existe demora en despachar pedidos.

Con el sistema informático, se podría evitar múltiples problemas tales como: que se agote la existencia de repuestos, el hurto de material y resolver la problemática de los códigos por repuesto nuevos el cual se lo generaría automáticamente, a su vez la de evitar duplicidad de información, como también de una accesibilidad a toda la información existente en el Área de Bodega; para brindar soluciones rápidas y acorde a las necesidades de la empresa.

Formulación del problema

Según la problemática que existe en la empresa surge la siguiente interrogante:

¿Cómo mejorar el desempeño de la empresa "CarPoint", con un sistema automatizado para el control del inventario en Bodega?

Justificación

Justificación Teórica

El desarrollo de software en la actualidad es una necesidad para las empresas el cual brinda un control rápido, preciso y automatizado sobre sus procesos para evitar el factor de error humano.

Justificación Práctica

Con el desarrollo que se desea implementar para la gestión de Bodega se logrará obtener un manejo adecuado sistematizado y organizado para la disponibilidad de información a todo momento y cuando se lo requiera.

Justificación Metodológica

Las metodologías que se implementaron son:

- Metodología de observación.- Mediante la observación seleccionamos los objetos que queremos analizar.
- Metodología inductiva.- Con el resultado de la metodología de observación se busca conclusiones parecidas a la aplica actualmente.
- Metodología deductiva.- Una vez realizado la observación llegar a la conclusión de estudio.
- Metodología de análisis.- Buscar la relación efecto-causa de la realidad del estudio.
- Metodología de síntesis.- Buscar explicaciones del estudio, de lo simple a lo más complejo.

Justificación Económica – Social

Con el sistema informático se tendrá un mejor control sobre el manejo de repuestos, evitando que existan diferencias de inventario siendo beneficioso para la empresa.

Objetivo General

Implementar un sistema informático de escritorio para la gestión del inventario de la empresa "CarPoint" bajo la metodología de desarrollo XP para tener un correcto control de producto de Bodega.

Objetivo específicos

- Analizar el proceso de Bodega actual de gestión de inventario dentro de la empresa "CarPoint".
- Diseñar la arquitectura de datos para el correcto proceso de Bodega.
- Codificar estrategias de solución para diseñar el proceso a nivel Bodega.
- Implementar el sistema de Bodega con el personal de la empresa.

1 CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

1.1 Taller Automotriz

Es el lugar donde ingresan automóviles de todo tipo de marcas; por lo general con fallas mecánicas donde el especialista en este caso en mecánica, quien con su conocimiento de cómo funciona el automotor; investiga cual puede ser su desperfecto para luego arreglarlo manualmente.

En el mercado automotriz existen talleres mecánicos para marcas específicas u otros que reparar todo automotores. Así mismo hay talleres que solo venden productos de su marca y otras que son independientes que reparan y venden todo tipo de marcas, son conocidas como "Multimarca". Existen también talleres que se especializan en diferentes partes de los vehículos, pero hay otros que revisan exclusivamente solo la parte mecánica del automotor. Algunos servicios que tienen los talleres mecánicos son:

- Desabolladura
- Pintura
- Accesorios
- Mecánica General
- Mecánica de alto nivel apoyado por tecnologías de punta.
- Importación y venta de repuestos legítimos.

1.2 Bodega

"*Nos referimos a un almacén de ciertas dimensiones, propios de la empresa, sin embargo los conceptos son completamente asimilables y utilizables para un almacén de tipo medio o pequeño, típico de las empresas denominadas pymes.*" (Tejero, 2008)

Este lugar está construido especialmente para almacenar para todo tipo de repuestos que requieren en el ámbito automotriz tales como son: aceites, filtros de aire, filtro de gasolina, bombillos, bujías, bandas de transmisión, líquidos de frenos, etc.

1.3 Inventario

"*Un inventario puede ser algo tan elemental como una botella de limpiador de vidrios empleada como parte del programa de mantenimiento de un edificio, o algo más complejo, como una combinación de materias primas y sub ensamblajes que forman parte de un proceso de manufactura.*" (Ballou, 2004)

El inventario en la parte fundamental de un negocio, porque es el encargado de manejar de una manera lo más posible exacta de los activos que tiene la empresa, el cual debería ser controlada de manera diario, semanal, trimestral o hasta anual.

1.4 Control de existencias

"Sirve como medio de control en el sistema de inventario permanente exigido para algunas empresas a nivel local. Sin embargo, con las NIIF se asume que todas las empresas deben aplicar el mismo sistema de control de inventarios, porque allí no se hace diferencia entre inventarios periódicos o permanentes." (Martínez & Celis, 2015)

Son también denominadas tarjetas kárdex, el cual consiste en anotar en hojas separadas, los registros de entrada, salida y el saldo por cada tipo de proceso; en los registros incluyen: materia prima, artículos para le venta y repuestos.

1.5 Kárdex

También conocido libro de almacén de los inventarios que ingresan o salen de un almacén el cual es registrado en un cuaderno, este contiene el número de manera precisa de cuantos hay en existencia o de faltantes.

La finalidad de este libro es de registrar los ingresos y egresos por cada producto al fin conocer su saldo final en un periodo contable. Este se lo puede por lo regularmente dos veces por año, el primero puede ser al finalizar el año fiscal y otro al cualquier mes de año que llamado inventario inicial.

"Por su aspecto legal y técnico este libro es obligatorio y principal de foliación simple (enumeración de folios) en el que se anotaran y registraran todos los inventarios que la

empresa realiza bajo su firma y responsabilidad como reflejo de todo lo que posee la empresa o negocio para su funcionamiento y desarrollo." (Picón, 1998)

1.6 Sistemas de Inventario

Existen tres tipos de inventario para el cálculo del costo de venta el permanente, el periódico:

Tabla 1 Tipos Sistemas de Inventario

Sistema de Inventario Periódico	Sistema de Inventario Contínuo
• Es el inventario físico al finalizar un periodo de tiempo o estados financieros. • Se lo realiza al menos anualmente. • Puede tener un alto costo de perdida para la empresa.	• Es actualizada por cada compra. • Provee mayo control interno sobre activos. • Produce información más oportunas

Fuente**:** (Métodos, técnicas y sistemas de valuación de inventarios. Un enfoque global, de José Alejandro Fuertes; 2015)

Elaborado por: EL Autor

1.7 Valorización de existencias del inventario

La valorización de existencias depende de cómo ha sido adquirida el producto, si es por compra a otra empresa, precio de adquisición, o fabricada, coste de producción.

Por cada clase de existencias se lleva una ficha donde se ingresan todos los movimientos o transacciones del almacén sin olvidar las existencias mínimas y máximas.

"Cada vez que se realiza una compra de existencias o una venta de existencias se tiene que anotar en la ficha de almacén, para poder tener un buen control de estas." (Finanzas y contabilidad Copyright, 2016)

1.8 Métodos de valorización de existencias

A continuación, se detalla en la tabla siguiente de comparación sobre los diferentes tipos de valorización de existencias:

Tabla 2 Valorización de existencias

FIFO	LIFO	PMP
Proviene de las siglas *First in – First out* el que primero entra es el primero en salir, en otras palabras la primeras que entran a Bodega son las primeras en vender.	Proviene de las siglas *Last in – First out* la última en entrar es la primera en salir, en otras palabras lo último en entrar en Bodega es lo primero en vender.	Es una media aritmética de los precios versus el número de productos adquiridos.

Fuente: (Joannés Vermorel, MÉTODO DE INVENTARIO, 2016)
Elaborado por: El Autor.

Imagen 1. Formula del Precio Medio Ponderado

$$PMP = \frac{\text{(existencias iniciales x valor) + (todas las unidades compradas o producidad x valor)}}{\text{existencias iniciales + todas las unidades compradas o producidas}}$$

Fuente: **(Finanzas y contabilidad Copyright © 2018, 2016)**

Cada vez que ingresa productos a la Bodega, se modifica el PMP y se reduce el valor de las existencias y cambia el valor de la salida del PMP.

1.9 Norma internacional de contabilidad 2 - NIC 2

El objetivo de esta Norma es prescribir el tratamiento contable de los inventarios, dentro del sistema de medición del costo histórico. Un tema fundamental en la contabilidad de los inventarios es la cantidad de costo que debe acumularse en un activo, para diferirlo hasta que los ingresos correspondientes sean reconocidos.

Esta norma suministra una guía práctica para la determinación de tal costo, así como para el subsecuente reconocimiento cómo gasto del periodo, incluyendo también cualquier deterioro que rebaje el importe en libros al valor neto realizable. También suministra una guía sobre las fórmulas de costo que se usan para calcular los costos de los inventarios.

1.10 Metodologías Agiles

En febrero del 2001 (UTA-EEUU) se da la idea del término ágil para el desarrollo de software, el cual el grupo que la fundo sin fines de lucro se llama THE AGILE ALLIANCE.

Este método nació por la necesidad de implementar software de manera rápida para desarrollar proyectos sin la necesidad de tanta documentación para el mismo.

1.11 Tipos de Metodologías Ágiles

Imagen 2 Metodologías Ágiles

Fuente: **(Bravent IT consulting company, 2015)**

1.11.1 Extreme Programming (XP)

Es uno de los métodos más usados para el desarrollo de software, es rápida, ligera, mucho más cómoda para generar software.

1.11.2 Scrum

Este método está basado en la orientación a objetos, este tiene la particularidad de que se reúnen por 30 días con el objetivo de desarrollarlo en ese tiempo, es también llamado de carrera. Cada día se reúnen dentro de los 15 a 30 minutos.

1.11.3 Dynamic Systems Development Method (DSDM)

Este método se trabaja directamente con el usuario de forma progresiva e iterativa.

En la siguiente figura se expone el porcentaje de los métodos más usados para el desarrollo de software.

1.12 Metodología Extreme Programming

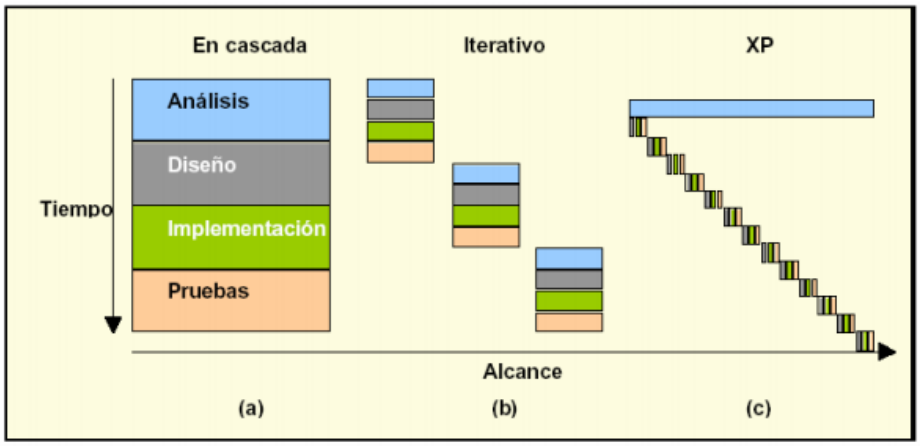
Es un método de desarrollo ágil, disciplinado con muchas soluciones sencillas, la característica de dicho método que se desarrolla mientras surgen cambios. Sus principales características se puede decir que son la de acortar ciclos de desarrollo e involucrar al cliente desde el principio hasta el final del proyecto.

"Extreme Programming (XP) surge como una nueva manera de encarar proyectos de software, proponiendo una metodología basada esencialmente en la simplicidad y agilidad. Las metodologías de desarrollo de software tradicionales (ciclo de vida en cascada, evolutivo, en espiral, iterativo, etc.) aparecen, comparados con los nuevos métodos propuestos en XP, como pesados y poco eficientes. La crítica más frecuente a estas metodologías "clásicas" es que son demasiado burocráticas. Hay tanto que hacer para seguir la metodología que, a veces, el ritmo entero del desarrollo se retarda. Como respuesta a esto, se ha visto en los últimos tiempos el surgimiento de "Metodologías Ágiles". Estos nuevos métodos buscan un punto medio entre la ausencia de procesos y el abuso de los mismos, proponiendo un proceso cuyo esfuerzo valga la pena." (Joskowicz, 2008)

1.13 Ciclo de visa de software en XP

A continuaciones algunas de los principales conceptos de desarrollo de software tradicionales:

Imagen 3 Metodologías de desarrollo de software



Fuente: (Joskowicz, 2008)

1.13.1 Modelo en cascada:

Son secuencias de actividades planificadas y bien estructuradas, es una de las metodologías más usadas y más extensas. Este consiste en que cada detalle se conoce de antemano incluso antes de desarrollar o codificar con esto se asumirá que no habrá más cambios a lo largo del ciclo de desarrollo.

1.13.2 Modelo incremental:

Consiste en que primero se desarrolla la arquitectura completa del sistema seguido de incrementos funcionales. Para continuar con cada ciclo de vida este siempre va a depender del anterior sin cambiar sus interfaces, si termina un ciclo no hay vuelta atrás.

1.13.3 Modelo espiral:

Es aquella que trata de combinar el modelo en cascada con el modelo evolutivo. Este modelo realiza el estudio de los riesgos del proyecto para prever varios ciclos o vueltas en espiral. Tiene cuatro etapas:

- Definición de objetivos.
- Evaluación y reducción del riesgo.
- Desarrollo y validación.
- Planificación del siguiente ciclo.

1.13.4 Modelo XP:

Este modelo se basa en las cuatro variables costo, tiempo, calidad y alcance de las cuales tres de ellas pueden ser fijas arbitrariamente por personas externas al grupo que la está desarrollando. Con esto los desarrolladores podrán determinar el tiempo que durará el proyecto.

La metodología XP tiene las siguientes reglas y prácticas:

- Planificación.
- Diseño.
- Desarrollo.
- Pruebas.

a) **Planificación** Es el dialogo entre el cliente, el programador y jefe del proyecto donde lo primero que es hacer las historias de usuarios con esto se puede evaluar

cuanto tiempo conlleva desarrollar cada una. Finalizado la historia de usuarios se lleva una pequeña reunión con las diferentes personas que integran el proyecto con la finalidad de llegar a un acuerdo de cronograma de tiempo para cada fase de iteraciones.

b) **Diseño:** Esto hace referencia a lo simple y claro de los diseños basado en los siguientes conceptos:

   La simplicidad de un diseño se implementa más rápido con que completo lo cual es hacer lo más simple y sencillo posible para que funcione.

   Las soluciones "spike" son cuando no se puede estimar un tiempo o si existe algunas dificultades o problemas técnicos se utiliza los llamados "spike" estos programas solo sirven para probar o evaluar.

   La recodificación el cual es escribir nuevamente el código del programa sin cambiar su funcionalidad, la metodología XP sugiere que se reutilice el código para no perder tiempo en hacer todo de nuevo y hacerlo lo más simple posible.

   Una metáfora deben ser utilizadas de las más claras y simples para el propósito del proyecto como finalidad guiar a la arquitectura. Las metáforas deben estar de acuerdo tanto el cliente como el grupo de desarrolladores para hablar todo el mismo idioma.

c) **Desarrollo:** Para llevar a cabo el desarrollo del código se debe regir a los siguientes requisitos:

   Disponibilidad de cliente: Es una parte esencial del proyecto para desarrollarlo y terminarlo de forma exitosa. Con esto se previene posibles problemas que pueda existir en el desarrollo del software y mientras más tiempo se pase con el usuario menos malos entendidos se tendrá durante el proyecto.

   Uso de estándares: XP tiene su programación basa en estándares para que sea entendida por todo el personal de desarrollo.

   Ritmo sostenido: Todo los desarrolladores deben llevar un tiempo sostenido de trabajo, todos deben llevar un ritmo constante y razonable para no sobrecargarse de trabajo, si el proyecto sobrepasa el tiempo establecido, se debe renegociar el plan de entrega realizando una reunión con los clientes y el equipo de desarrolladores.

d) **Pruebas:** Antes de ser liberados o publicados a los usuarios deben pasar por las pruebas unitarias el cual consiste en ser llevado a un "Test-driven programming" para poder corregir, cambiar o recodificar parte del mismo.

   Las detecciones y correcciones de errores deben ser corregidas inmediatamente y teniendo en cuenta que no se vuelva a ocurrir, una vez corregido todos los errores es necesario volver a ser sometido a pruebas.

   Las pruebas de aceptación son la comprobación de las historia de usuario ha sido correctamente implementada, el cual se realiza pruebas de caja negra. El cliente es el responsable de verificar que los resultados de las pruebas sean correctas. Las historias de usuarios no se pueden dar por finalizadas sin haber pasado las pruebas de aceptación.

1.14 Scrum

En 1993 nace Scrum el cual viene de un cambio radical de manera de hacer proyectos de software porque antes de esta fecha tomaba mucho tiempo en realizarlos de meses hasta años en culminar el proyecto.

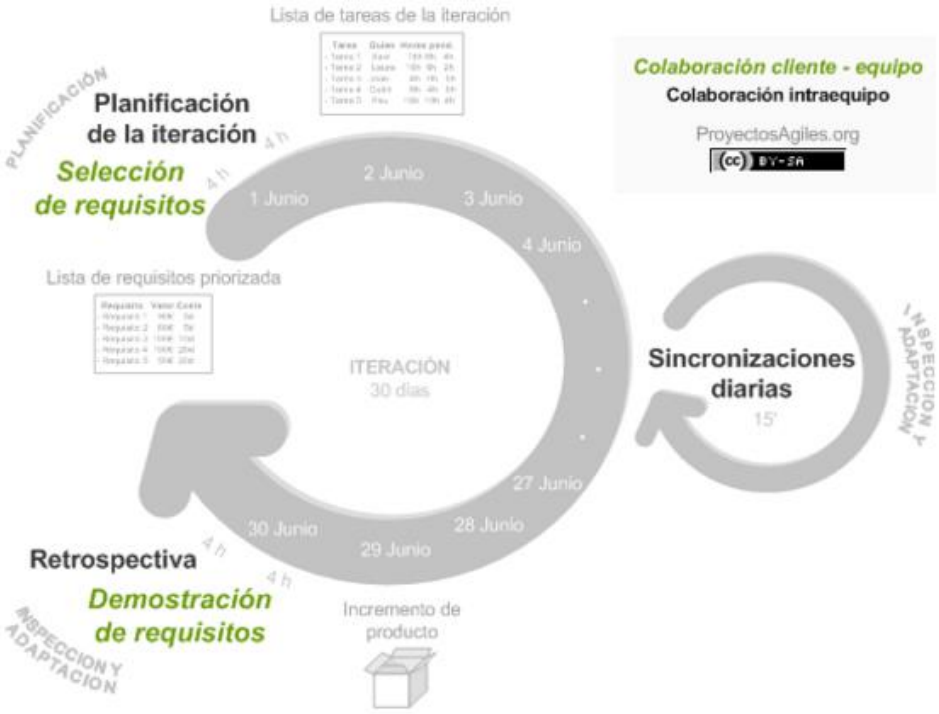
En la actualidad es usa en varias empresas de renombre como por ejemplo Toyota, FBI en el ciclo de vida de combate, etc.

"*En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales. "* (Albaladejo, 2015)

Scrum es utilizado cuando al cliente no se le está entregando lo solicitado, cuando toma demasiado tiempo en entregar el resultado, también se utiliza cuando es de baja calidad o no es aceptable, cuando se necesita reacción frente a competencias, cuando el equipo no está motivado y existe alta rotación, al identificar carencias sistemáticas y sirve también para ser utilizado en procesos de desarrollo especializado.

El equipo es de 3 a 4 personas en un periodo de normalmente de 2 semanas aunque puede prolongarse en 4 semanas, por cada ciclo se tiene que proporcionar un resultado para que el usuario pueda revisarlo.

Imagen 4 Ciclo de vida de Scrum



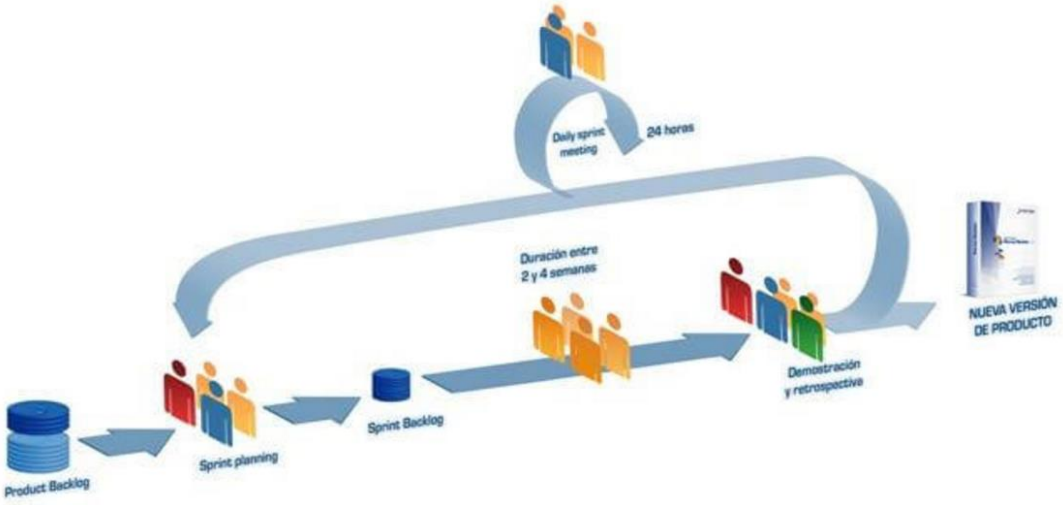
Fuente: (Albalejo, 2015)

Scrum tiene algunos procesos que se detallan a continuación:

I. **Product Backlog.-** Son conjunto de historias que se encuentran en un lenguaje no técnico considerando su beneficio y coste, durante el proceso del proyecto se van ajustando al proyecto.
II. **Sprint.-** Es el periodo donde trabajan para convertir las historias en product backlog.
III. **Sprint Planning.-** Es una reunión donde se presentan los sin números de historias y cuantas van a ser complementadas para realizar sprint.
IV. **Sprint Backlog.-** Lista de tareas para llevar la historia de sprint.
V. **Daily sprint meeting.-** Son reuniones diarias de 15 minutos donde cada uno expone lo realizado.

VI. **Demo y retrospectiva.-** Son demostraciones de las historias la cual tienen que indicar lo realizado.

Imagen 5 Procesos y Roles de Scrum



Fuente: **(SOFTENG, 2011)**

1.15 Roles

Es un equipo de personas enfocado en el desarrollo de software, definir las características a desarrollar y solventar todos los problemas que se puedan presentar durante el modelamiento del proyecto.

El equipo está formado por las siguientes personas:

a) **Scrum master.-** Es el encargado del proyecto, su trabajo es de dirigir y hacer cumplir las reglas junto con Product Owner
b) **Product Owner.-** Representantes que usan el software como son los clientes y accionistas, el cual es el responsable del ROI del proyecto, está encargado de transmitir la misión del proyecto.
c) **Team.-** Son las personas técnicas profesionales capaces de desarrollar de manera conjunta con la ayuda de las historias.

En conclusión se escoge la metodología ágil XP (Extreme Programming) porque se lo realiza de manera ágil y rápida para el desarrollo de software, esta metodología también se basa mucho con la comunicación con el cliente durante todo el desarrollo del proyecto, tratando de satisfacer las necesidades del mismo, a su vez usando la simplicidad al momento de

desarrollar todas sus funcionalidades, cabe recalcar que el cliente es la persona que verifica y aprueba el desarrollo de cada fase.

1.16 Herramientas de Licenciamiento Open Source

Es poder realizar software o utilizar todos sus utilitarios para poder crear nuevos programas sin necesidad de tener licencias de uso.

"Cuando un programa de software libre deja de estar en manos de su autor, esto no significa necesariamente que siga siendo software libre para cualquiera que se haga con una copia de él. Por ejemplo, el software de dominio público —software sin copyright— es software libre, pero cualquiera puede modificarlo y hacer una versión propietaria a partir de él." (Stallman, 2004)

Gracias a Open Source en la actualidad se puede acceder a cualquier tipo de producto, como es el ejemplo de Linux y Apache, cuyos proyectos no le pide favores a nadie.

Con esto se puede desarrollarlo, copiarlo, modificarlo o hasta mejorarlo gracias a la tecnología libre.

1.17 Java

Java surgió en 1991 cuando un grupo de ingenieros de Sun Microsystems trataron de diseñar un nuevo lenguaje de programación destinado a electrodomésticos. La reducida potencia de cálculo y memoria de los electrodomésticos llevó a desarrollar un lenguaje sencillo capaz de generar código de tamaño muy reducido.

Debido a la existencia de distintos tipos de CPUs y a los continuos cambios, era importante conseguir una herramienta independiente del tipo de CPU utilizada. Desarrollaron un código "neutro" que no dependía del tipo de electrodoméstico, el cual se ejecutaba sobre una "máquina hipotética o virtual" denominada Java Virtual Machine (JVM).

"Era la JVM quien interpretaba el código neutro convirtiéndolo a código particular de la CPU utilizada. Esto permitía lo que luego se ha convertido en el principal lema del lenguaje: "Write Once, Run Everywhere". A pesar de los esfuerzos realizados por sus creadores,

ninguna empresa de electrodomésticos se interesó por el nuevo lenguaje." (García de Jalón, y otros, 2000)

Java a sus inicios solo era un lenguaje de programación para las computadoras, pero a los finales de 1995, con la incorporación del programa NetScape Navigator, causan revuelo en el Internet con su versión Java 1.1. En 1997 aparece mejorado la versión 1,2 que después la renombraron como Java 2 en el año 1998.

Es un lenguaje sofisticado para programar, es uno de los de los lenguajes de programación más usados en la actualidad tanto para crear software de escritorio como también para la android, es un eficaz y muy bueno para trabajar en el modelo orientado a objetos.

"*Es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más*." (Oracle, 2014)

1.18 Netbeans

Es un entorno de desarrollo libre, éste programa está basado en el lenguaje Java, el cual contiene innumerables módulos para las distintas áreas de la programación. A demás es un producto gratuito y libre.

En la actualidad es unos de los programas más utilizados para el desarrollo de software, siendo así uno de los pioneros en proyectos de código abierto. También existe alta demanda de desarrolladores para este tipo de lenguaje.

"NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos (Actualmente Sun Microsystems es administrado por Oracle Corporation)". (Sanchez Allende, Fernandez Toribio, & Moreno Díaz, 2005)

1.19 PostgreSQL

Michael Stonebraker inicio el prototipo en los años 80's lo llamaron Post Ingres, su finalidad fue solucionar la carencia que una base de datos para la época, ya que en ese tiempo existían muchos problemas con los programas que administraban las bases de datos. En la actualidad es considerado el gestor de bases de datos más avanzados que existe en el mercado.

Es un gestor de Base de datos que es de software libre, es muy utilizado en la actualidad en el campo de desarrollo de software por el mismo hecho de que no se tiene que pagar por licencia para utilizar el programa.

Es compatible con cualquier programa de lenguaje de programación fácil de usar y no muy ambiguo en su uso.

Cabe mencionar que es gratuito esté pertenece al grupo de open-source, siendo unos de los más populares en la actualidad, como fue en su época MySql, que ahora se encuentra en propiedad de Oracle y tiene muchas limitantes.

"Una de las características interesantes de PostgreSQL es el control de concurrencias multiversión; o MVCC por sus siglas en inglés. Este método agrega una imagen del estado de la base de datos a cada transacción. Esto nos permite hacer transacciones eventualmente consistente, ofreciéndonos grandes ventajas en el rendimiento." (PostgreSQL, 2015)

1.20 Arquitectura MVC

Es un diseño de programación que separa el código de programa con las distintas responsabilidades mediante la interfaces de usuarios.

En la arquitectura MVC se potencializa la reutilización de código, separación de conceptos y facilidad de mantenimiento.

La arquitectura está basada en tres capas diferentes:

- Modelo (Model)
- Vistas (Views)
- Controladores (Controllers)

1.20.1 Modelo

Está capa es donde se encuentra los datos el cual obtendrá la información de la base de datos el cual se puede realizar las diferentes opciones como son select, insert, delete, update, etc. Adicional se maneja los accesos basados en clases y objetos.

1.20.2 Vistas

Es la interfaz gráfica de usuario que genera el código de programación para los diferentes estados de nuestra aplicación.

1.20.3 Controladores

Son acciones que solicita la aplicación, como buscar información, actualizar datos, visualizar un dato, etc.

Está capa es la encargada de enlazar las vistas con los modelos, sin embargo no manipula los datos solo implementa las múltiples necesidades del software.

2 CAPÍTULO II. PROPUESTA

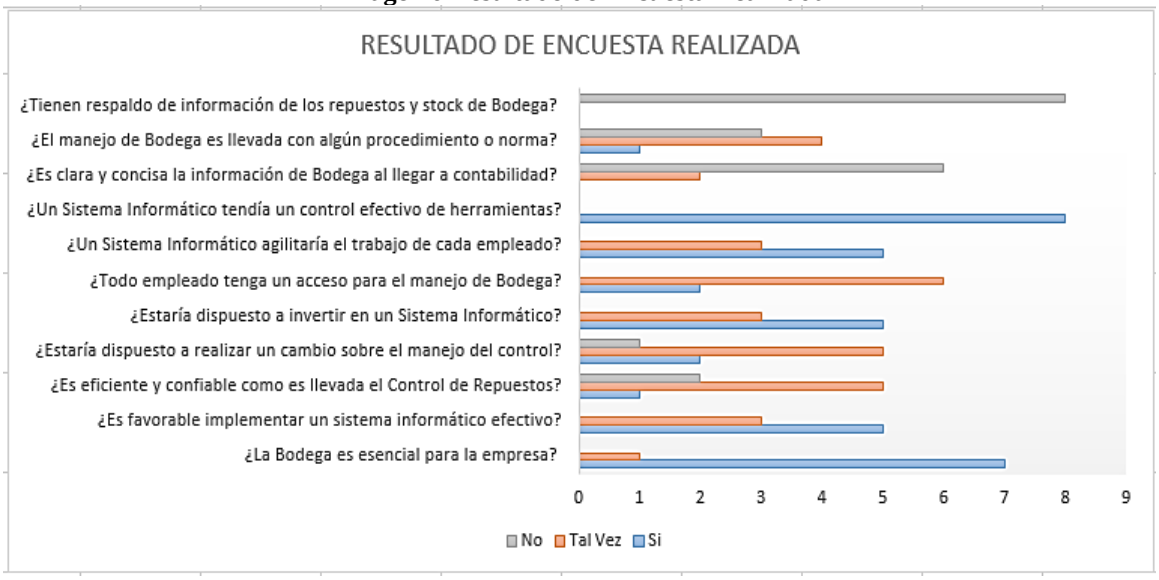
2.1 Recopilación de información

2.1.1 Encuesta

Se realizó una entrevista a dos inversionistas de la empresa CarPoint, a 3 personas profesionales (Jefe de taller, 2 mecánicos) y a 3 empleados (2 ayudantes y al encarga de la Bodega), las preguntas con sus respuestas los encontrará en los Anexos 1 y 2.

A continuación el resultado de la encuesta:

Imagen 6 Resultado de Encuesta Realizada



Fuente: Propia
Elaborado por: El Autor.

El objetivo de la encuesta realizada es la de conocer si es factible implementar un sistema informático para la empresa “CarPoint” para el manejo del inventario de bodega.

Como observación se toma en cuenta la pregunta 8 el cual indica que el proceso de inventario sea llevado bajo un sistema informático, por el cual se tiene el apoyo total por parte de la gerencia como también por parte de los inversionistas.

En conclusión:

Según los resultados de la encuentra realizada se evidencia el total apoyo al cambio a un sistema informático para el manejo adecuando de la Bodega del taller; con la generación de códigos de forma automática se solucionaría una gran problemática que existe entre personal y de optimización de tiempo en inventarios.

2.1.2 Observación

Se acude a la empresa “CarPoint” cada sábado desde las 08:00am hasta las 13:00 con el propósito de conocer como son llevados los procesos, con la finalidad de recopilar información que permita desarrollar el sistema informático.

Se pudo observar las múltiples necesidades que requiere la empresa de un sistema informático que permita llevar un adecuado manejo de inventario de los artículos en Bodega.

Se evaluó la situación de cómo es llevado los diferentes procesos por área de trabajo dentro de la empresa, prestando más atención en la parte de Bodega, en donde se pudo evidenciar que no tienen levantados procesos de manejo de inventario.

Con la colaboración de la gerencia, inversionistas y resto de empleados se puede considerar que la información proporcionada es confiable y veraz.

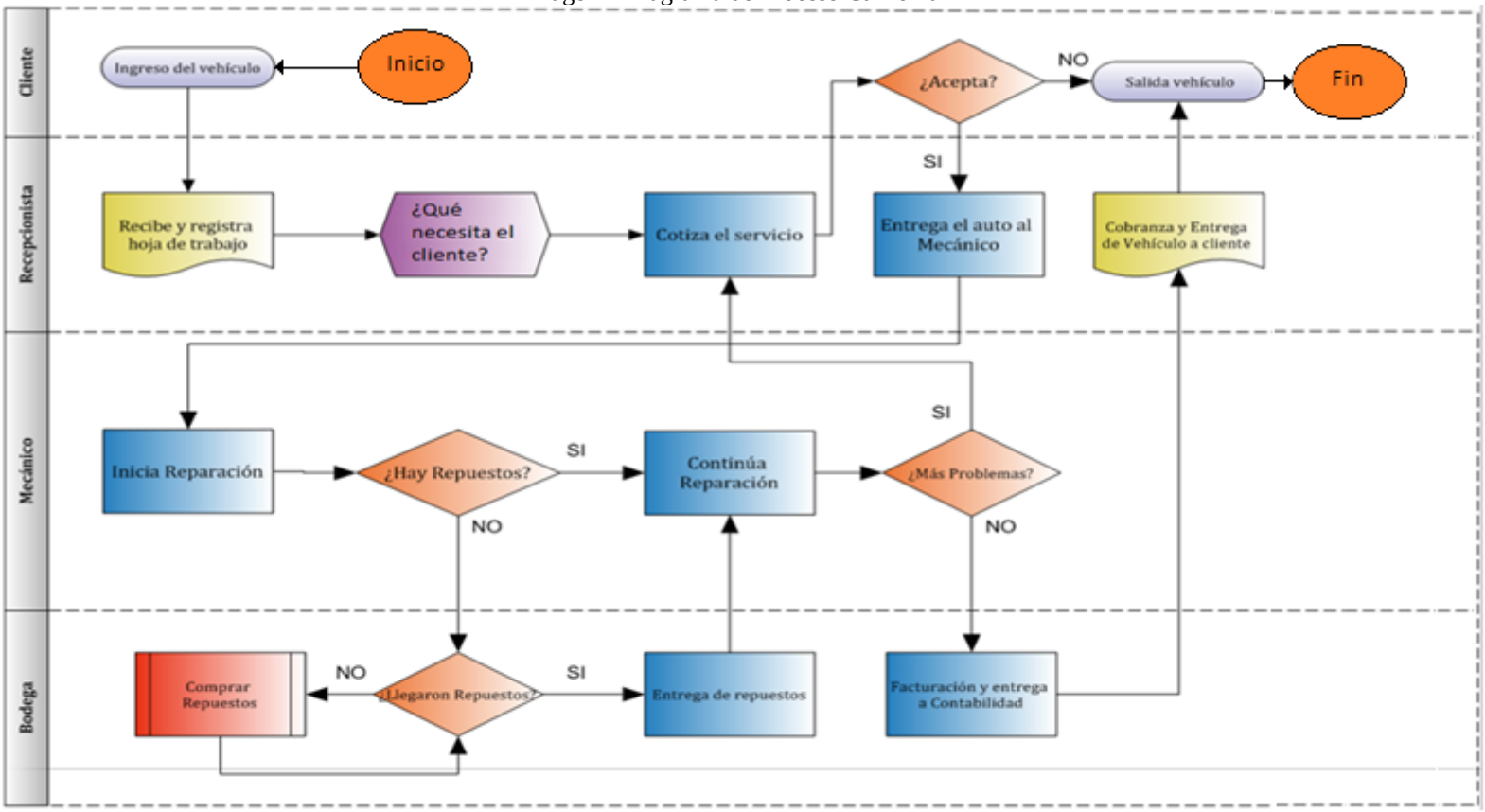
Estos dan como resultado los siguientes diagramas de procesos levantados que se detallan en el siguiente capítulo.

2.2 Diagramas de Procesos

2.2.1 Proceso Global

El siguiente diagrama se lo desarrollo en base a la observación sobre cómo es llevado sus procesos en la empresa “CarPoint”.

Imagen 7 Diagrama de Proceso CarPoint

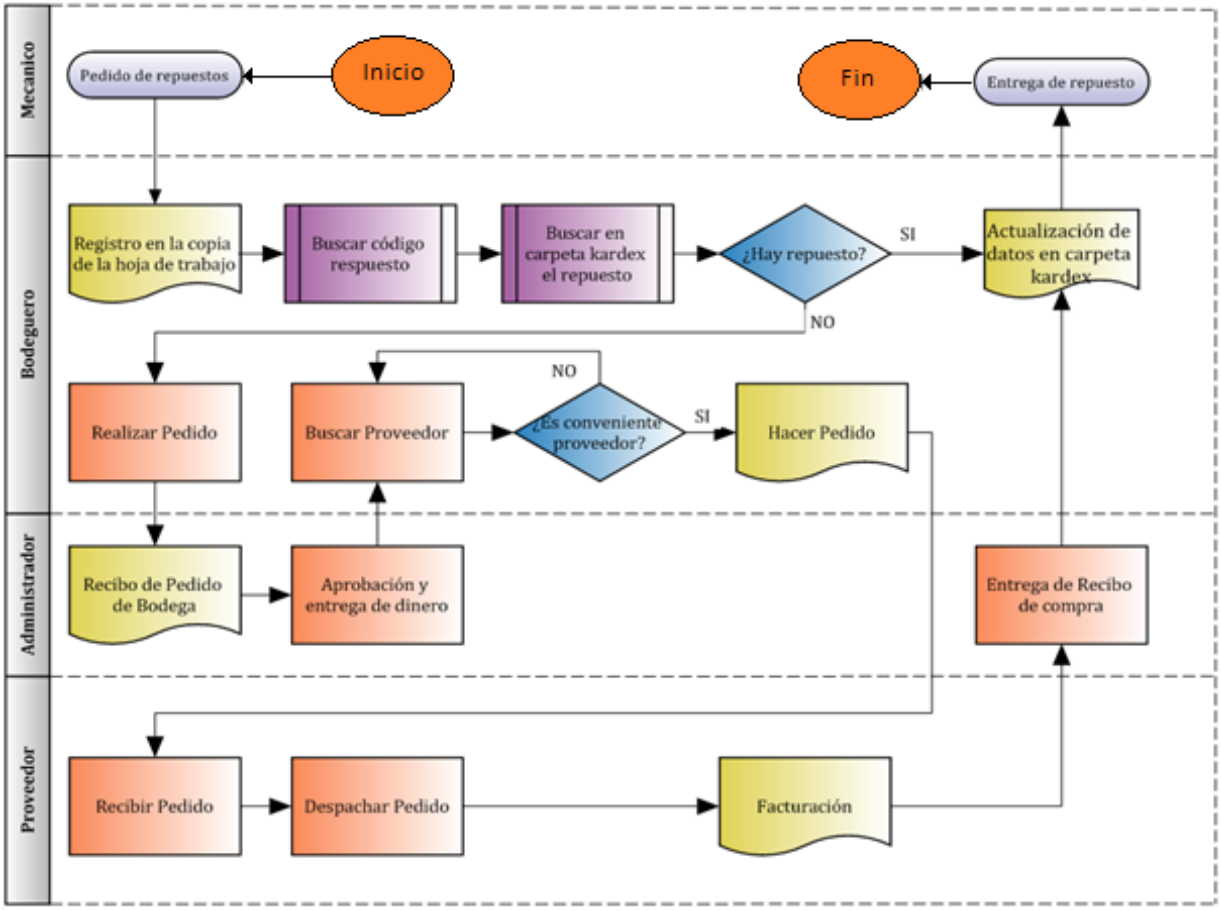


Fuente: Propia.
Elaborado por: El Autor

2.2.2 Proceso Actual Bodega

Diagrama de proceso actual de Bodega en la actualidad.

Imagen 8 Proceso de almacenamiento de repuestos

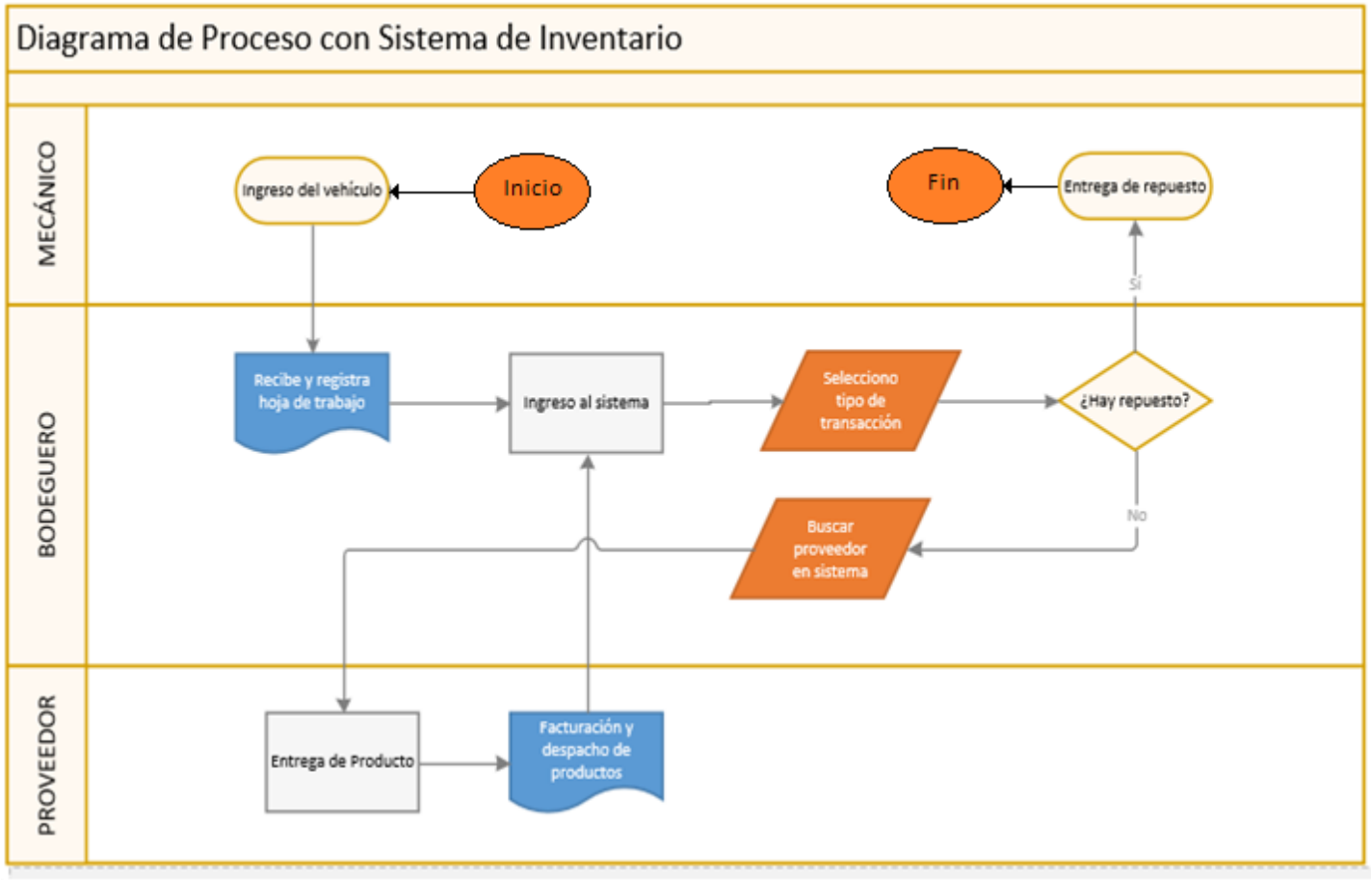


Fuente: Propia
Elaborado por: El Autor.

2.2.3 Proceso Propuesto

En el siguiente diagrama es el proceso de almacenamiento propuesto luego de automatizar los procesos manuales.

Imagen 9 Proceso de Almacenamiento Propuesto



Fuente: Propia
Elaborado por: El Autor.

2.3 Especificación de Requerimientos

2.3.1 Ámbito del Software

El nombre del programa es “SafePoint” ya que la empresa se llama CarPoint y Safe significa resguardo en este caso resguardo de repuestos de la empresa CarPoint.

En la actualidad se lleva la gestión de inventario de manera manual el cual consiste en que llega el automotor, es anotado en la hoja de trabajo, examina el técnico automotriz pide a Bodega el repuesto, el cual con una segunda hoja es anotado los repuestos que son entregados y firmado al técnico. El encargado de Bodega, según el repuesto indicado, busca en una hoja llena de códigos, una vez identificado el código anota el repuesto en la carpeta llamada Kárdex, aquí se puede visualizar el tipo de material, el stock y el precio. Dicho precio es colocado según el criterio del encargado de Bodega.

El objetivo del proyecto es la de tener un software eficiente y a la altura de la empresa dejando atrás el tener archivado en hojas todo el movimiento de repuestos. Está enfocado en tener un excelente y confiable manejo del inventario de Bodega.

El sistema informático “SafePoint” tiene las siguientes funcionalidades:

a) Registro de Artículo
b) Registro de Clientes
c) Registro de Proveedores
d) Módulo de Movimientos
e) Módulo de Maestro
f) Módulo de Administración
g) Consulta de movimientos de stock
h) Reportes

El programa no está contemplado para resolver toda las problemáticas que existe dentro de la empresa, será netamente para el manejo de Bodega específicamente de repuestos.

2.3.2 Funciones del producto

Mediante la historia de usuarios podemos obtener las funciones principales que se llevara a cabo en el proyecto bajo la metodología Programación Extrema XP.

A continuación un resumen de las historias de usuarios con la asignación de iteración

Tabla 3 Resumen de Historia de Usuarios

N°	Nombre	Prioridad Desarrollo	Prioridad Negocio	Iteración Asignada
1	Administración de usuarios	Alto	Alto	1
2	Ingreso al Sistema SafePoint	Medio	Alto	2
3	Gestionar Artículos	Alto	Alto	3
4	Gestionar Clientes	Alto	Alto	4
5	Gestionar Proveedores	Alto	Alto	5
6	Gestionar Venta	Alto	Alto	6
7	Gestionar Compra	Alto	Alto	7
8	Gestionar Devolución	Alto	Alto	8
9	Generar Reporte	Medio	Alto	9

Fuente: Propia.
Elaborado por: El Autor

2.3.3 Historia de usuarios

Tabla 4 Historia de usuarios HU1

HISTORIA DE USUARIOS			
Usuario:	Administrador	**ID:**	**HU1**
Nombre historia:	Administración de usuarios		
Programador Responsable:	Edison Martin Molina Soria		
Riesgo de desarrollo:	Alto	**Prioridad del negocio:**	Alto
Descripción:	• El administrador será el único que podrá gestionar los usuarios y catalogar por Perfiles.		
Observaciones:	• El administrador será el único que tendrá permisos para todos los módulos.		

Fuente: Propia.
Elaborado por: El Autor

Tabla 5 Historia de usuario HU2

HISTORIA DE USUARIOS			
Usuario:	Administrador	**ID:**	**HU2**
Nombre historia:	Registro de compra y venta		
Programador Responsable:	Edison Martin Molina Soria		
Riesgo de desarrollo:	Medio	**Prioridad del negocio:**	Alto
Descripción:	• El Administrador y el Bodeguero tendran asignado con usuario y contraseña para registrar los detalles que realiza en cada tipo de proceso de compra y venta.		
Observaciones:			

Fuente: Propia.
Elaborado por: El Autor

Tabla 6 Historia de usuario HU3

HISTORIA DE USUARIOS			
Usuario:	Administrador	**ID:**	**HU3**
Nombre historia:	Gestionar Artículos		
Programador Responsable:	Edison Martin Molina Soria		
Riesgo de desarrollo:	Alto	**Prioridad del negocio:**	Alto
Descripción:	El administrador deberá registrar, actualizar o eliminar un artículo, al registrar un artículo se requiere ingresar: • Aplicación • Descripción • Detalle • Costo • Precio de Venta		
Observaciones:	• La información ingresada es necesaria al realizar una venta o compra.		

Fuente: Propia.
Elaborado por: El Autor

Tabla 7 Historia de usuario HU4

HISTORIA DE USUARIOS			
Usuario:	Bodeguero	**ID:**	**HU4**
Nombre historia:	Gestionar Clientes		
Programador Responsable:	Edison Martin Molina Soria		
Riesgo de desarrollo:	Medio	**Prioridad del negocio:**	Alto
Descripción:	Los clientes deberá ser registrada, actualizada o eliminada por el bodeguero, al registrar un cliente se requiere ingresar: • Nombre • Apellido • Cédula • Dirección • Teléfono		

	• Email
Observaciones:	• El administrador podrá también gestionar clientes.

Fuente: Propia.
Elaborado por: El Autor

Tabla 8 Historia de usuario HU5

HISTORIA DE USUARIOS			
Usuario:	Administrador	**ID:**	**HU5**
Nombre historia:	Gestionar Proveedores		
Programador Responsable:	Edison Martin Molina Soria		
Riesgo de desarrollo:	Alto	**Prioridad del negocio:**	Alto
Descripción:	Los proveedores deberá ser registrada, actualizada o eliminada por el administrador, al registrar un proveedor se requiere ingresar: • Empresa • Ruc • Dirección • Teléfono • Email • Detalle		
Observaciones:			

Fuente: Propia.
Elaborado por: El Autor

Tabla 9 Historia de usuario HU6

HISTORIA DE USUARIOS			
Usuario:	Bodeguero	**ID:**	**HU6**
Nombre historia:	Gestionar Venta		
Programador Responsable:	Edison Martin Molina Soria		
Riesgo de desarrollo:	Alto	**Prioridad del negocio:**	Alto
Descripción:	• Permitirá registrar, modificar y eliminar la información de venta, junto con los datos de clientes y artículos seleccionados.		
Observaciones:	• El Bodeguero registra las ventas según las necesidades del cliente.		

Fuente: Propia.
Elaborado por: El Autor

Tabla 10 Historia de usuario HU7

HISTORIA DE USUARIOS			
Usuario:	Bodeguero	**ID:**	**HU7**
Nombre historia:	Gestionar Compra		
Programador Responsable:	Edison Martin Molina Soria		
Riesgo de desarrollo:	Alto	**Prioridad del negocio:**	Alto

Descripción:	• Permitirá registrar, modificar y eliminar la información de compra, junto con los datos de proveedores y artículos seleccionados.
Observaciones:	• El Bodeguero registra las compras según las necesidades del cliente.

Fuente: Propia.
Elaborado por: El Autor

Tabla 11 Historia de usuario HU8

HISTORIA DE USUARIOS			
Usuario:	Administrador	**ID:**	**HU8**
Nombre historia:	Gestionar Devolución		
Programador Responsable:	Edison Martin Molina Soria		
Riesgo de desarrollo:	Alto	**Prioridad del negocio:**	Alto
Descripción:	• Permitirá registrar, modificar y eliminar la información de compra o venta realizadas.		
Observaciones:	• El Administrador será el único encargado de realizar las devoluciones.		

Fuente: Propia.
Elaborado por: El Autor

Tabla 12 Historia de usuario HU9

HISTORIA DE USUARIOS			
Usuario:	Administrador	**ID:**	**HU9**
Nombre historia:	Generar Reportes Administrativos		
Programador Responsable:	Edison Martin Molina Soria		
Riesgo de desarrollo:	Medio	**Prioridad del negocio:**	Alto
Descripción:	• El administrador podrá visualizar reportes acerca de las ventas, compras, devoluciones, stock y Kárdex.		
Observaciones:	• El administrador con los reportes puede tomar decisiones de negocio.		

Fuente: Propia.
Elaborado por: El Autor

Tabla 13 Historia de usuario HU10

HISTORIA DE USUARIOS			
Usuario:	Bodeguero	**ID:**	**HU10**
Nombre historia:	Generar Reportes para Bodega		
Programador Responsable:	Edison Martin Molina Soria		
Riesgo de desarrollo:	Medio	**Prioridad del negocio:**	Alto
Descripción:	• El Bodeguero podrán visualizar reportes acerca de las ventas y compras realizadas.		
Observaciones:			

Fuente: Propia.
Elaborado por: El Autor

2.3.4 Características de los usuarios del sistema

El sistema informático será usado por los siguientes usuarios:

- Administrador: Persona encargada de la empresa con gran manejo del software previamente capacitado, su función es de creador de perfiles con sus respectivos usuarios y visualizar el status de la Bodega mediante reportes.

- Bodeguero: Persona encargado del inventario de los repuestos, su función es la de ingresar y despachar repuesto. Todo debe ser ingresado en el software.

- Programador: Persona encargada de realizar el software, su función es la de realizar un programa que satisfaga las necesidades de la empresa para el manejo de Bodega.

2.3.5 Restricciones

a. Políticas regulatorias

El software será desarrollado por código abierto lo que quiere decir software libre el cual no se tendrá que pagar por el uso del mismo, según los leguajes de programación Java y el motor de base de datos PostgreSQL.

b. Limitaciones de hardware

Para que se pueda implementar el software es necesario contar con un equipo computacional donde pueda funcionar, el cual se instalará los programas para su funcionamiento.

c. Funciones de control

Cada usuario con sus respectivos permisos ya establecidos por el administrador, puede acceder al sistema dependiendo de su rol para sus diferentes funciones.

d. Lenguaje del programa

El lenguaje estará hecho en español para la comodidad de los usuarios del taller CarPoint.

e. Información fiable

La información que genere el software será confiable sin manipulaciones al rato de presentar los reportes de Bodega.

f. Seguridad del software

El acceso es únicamente y exclusivamente con sus usuarios y contraseña el cual estarán encriptados en la base de datos. Y solo se podrá ejecutar en una sola terminal.

2.3.6 Requisitos

a) **Requerimientos funcionales**

El sistema informático tiene como objetivo los siguientes puntos funcionales:

RF01.- El sistema informático se ingresará con un usuario y contraseña autorizada.

RF02.- El sistema indicará cuadros informativos de los stocks actuales de Bodega.

RF03.- El sistema realizará ventas con los datos de clientes y artículos vendidos.

RF04.- Al generar una venta no se podrá ingresar un número de orden ya ingresado anteriormente.

RF05.- En Ventas solo desplegará los repuestos que tengan stocks mayores a cero.

RF06.- En Ventas validará que la cantidad sea mayor o igual a la que se tiene en stock.

RF07.- En Ventas y Compra no se podrá escoger dos veces el mismo repuesto.

RF08.- En Ventas antes de finalizar el pedido se podrá eliminar repuestos seleccionados.

RF09.- En Ventas antes de finalizar el pedido se podrá eliminar repuestos seleccionados.

RF10.- El sistema realizará compras con los datos del proveedor y artículos comprados.

RF11.- Los reportes se podrán generar ingresando el rango de fechas requerida.

RF12.- Los campo monto, total, subtotal, precio acepta valores con dos decimales.

RF13.- La fecha por cada venta está por defecto la del día actual sin poder cambiarla.

RF14.- Los campos nombre, apellido acepta caracteres únicamente alfabéticos.

RF15.- Los campos dirección, correo, acepta caracteres únicamente alfanuméricos.

RF16.- El producto será seleccionando de una lista pre-establecida por el administrador.

RF17.- Al crear un nuevo cliente, valida que no exista un numero de cedula ya ingresada.

RF18.- Al crear un nuevo proveedor, valida que no exista un número de ruc ya ingresada.

RF19.- Al crear un artículo, el código se generará con los tres primeros caracteres de Aplicación más Descripción más el número de código.

RF20.- Solo los Administradores podrán eliminar, actualizar y crear: usuarios, proveedores, stocks y devoluciones.

RF21.- Solo Administradores podrán cambiar los parámetros del sistema.

b) Requerimientos no funcionales

RNF01.- El sistema se podrá utilizar en sistemas operativos Windows.

RNF02.- Constante actualización con la base de datos por cada transacción.

RNF03.- El sistema enmascara la clave al ingresar en login, se encontrará encriptado con formato MD5.

RNF03.- Los permisos de usuarios serán gestionados por el administrador de la empresa.

RNF04.- Proceso de respaldos cuando el usuario lo requiera.

RNF05.- El programa es de sencillo aprendizaje en menos de 1 día se puedo manejar.

RNF06.- Los mensajes de error serán entendibles al usuario.

RNF07.- Se contara con un manual de usuario de fácil entendimiento.

RNF08.- La aplicación no puede ocupar más de 2 GB de espacio.

RNF09.- Requerimiento mínimo de memoria RAM será de 2 GB.

RNF10.- Sera desarrollado la aplicación con el método CamelCase.

RNF11.- El sistema se basara a las reglas de software libre GNU código abierto.

3 CAPÍTULO III. IMPLEMENTACIÓN

3.1 Diseño general

"Las cosas más sencillas podría funcionar" para lo cual conlleva una buena comunicación, sencillez, aprendizaje y seguridad, con esto se puede mejorar y arreglarlo mucho más rápido al momento de recodificar así como la duplicidad de código.

En esta fase en donde se define el proceso de automatización propuesta en el proyecto, el cual se va a basar en la manera que XP propone.

De acuerdo a la metodología se recopilaron las tarjetas CRC, las cuales permiten identificar las clases, detalles, sus responsables con sus colaboradores para conseguir un objetivo.

Al trabajar en conjunto con el cliente se garantiza que el producto final concluya, al mismo tiempo que se realiza correcciones por cada iteración.

3.1.1 Tarjetas CRC

Las tarjetas CRC son las que permiten tener una idea de la orientación a objetos, también que se relaciones las diferentes tipos de clases que existan.

Tabla 14 Tarjeta CRC Usuario

Nombre de Clase: Usuario	
Detalle: Gestión de usuarios	
Responsabilidades: • Crear usuario • Actualizar usuario • Eliminar usuario • Estado (Activo - Inactivo) • Consultar usuario	**Colaboradores:** • Perfiles

Fuente: Propia.
Elaborado por: El Autor

Tabla 15 Tarjeta CRC Perfiles

Nombre de Clase: Perfiles	
Detalle: Esta clase es la encargada de dar permisos a diferentes opciones del sistema según al perfil que corresponden.	
Responsabilidades:	**Colaboradores:**

• Crear Perfiles • Actualizar Perfiles • Eliminar Perfiles • Consultar Perfiles • Muestra Perfiles	• Usuario

Fuente: Propia.
Elaborado por: El Autor

Tabla 16 Tarjeta CRC Clientes

Nombre de Clase: Clientes	
Detalle: Esta información se llena al ingresar el cliente	
Responsabilidades: • Ingresar datos de nuevo Cliente • Modificar datos de Cliente • Eliminar datos de Cliente	**Colaboradores:** • Productos • Venta • Reporte

Fuente: Propia.
Elaborado por: El Autor

Tabla 17 Tarjeta CRC Proveedor

Nombre de Clase: Proveedor	
Detalle: Esta información ingresa el usuario con los datos de la factura de compra	
Responsabilidades: • Ingresar datos de nuevo Proveedor • Modificar datos de Proveedores • Eliminar datos de Proveedores	**Colaboradores:** • Repuestos • Compra • Reporte

Fuente: Propia.
Elaborado por: El Autor

Tabla 18 Tarjeta CRC Artículo

Nombre de Clase: Artículo	
Detalle: Esta información ingresa el usuario con los datos de la factura de compra	
Responsabilidades: • Ingresar datos de nuevo Artículo • Modificar datos de Artículo • Eliminar datos de Artículo	**Colaboradores:** • Proveedor • Compra • Venta • Reporte • Stock

Fuente: Propia.
Elaborado por: El Autor

Tabla 19 Tarjeta CRC Venta

Nombre de Clase: Venta	
Detalle: Esta información ingresa el usuario de bodega por cada despacho de artículos	
Responsabilidades: • Ingresar datos del Cliente • Ingresar Artículos • Guardar Venta	**Colaboradores:** • Cliente • Artículos • Reporte • Stock

Fuente: Propia.
Elaborado por: El Autor

Tabla 20 Tarjeta CRC Compra

Nombre de Clase: Compra	
Detalle: Esta información ingresa el usuario de bodega por cada compra de artículos	
Responsabilidades: • Ingresar datos del Proveedor • Ingresar Artículos • Guardar Compra	**Colaboradores:** • Proveedor • Artículos • Reporte • Stock

Fuente: Propia.
Elaborado por: El Autor

Tabla 21 Tarjeta CRC Reporte

Nombre de Clase: Reporte	
Detalle: Detalle de los diferentes tipos de procesos	
Responsabilidades: • Generar Reporte por fechas	**Colaboradores:** • Venta • Compra • Devoluciones • Cliente • Proveedor • Artículo • Stock

Fuente: Propia.
Elaborado por: El Autor

Tabla 22 Tarjeta CRC Devoluciones

Nombre de Clase: Devoluciones	
Detalle: Artículos que son devueltos o se dan de baja.	
Responsabilidades:	**Colaboradores:**

• Dar de baja • Regresar producto	• Venta • Compra • Cliente • Proveedor • Artículo • Reporte

Fuente: Propia.
Elaborado por: El Autor

Tabla 23 Tarjeta CRC Stock

Nombre de Clase: Stock	
Detalle: Cantidad de artículos en bodega.	
Responsabilidades: • Stock Máximo • Stock Mínimo • Stock Actual	**Colaboradores:** • Venta • Compra • Artículo • Reporte

Fuente: Propia.
Elaborado por: El Autor

3.1.2 Pruebas de aceptación

Con las pruebas de aceptación se verifica el correcto funcionamiento y que el usuario determine si es aceptable el sistema informático tanto en rendimiento como en funcional.

Tabla 24 Prueba de aceptación PA1

PRUEBA DE ACEPTACIÓN			
Nombre de la prueba:	Inicio de sesión al sistema	**ID:**	**PA1**
Responsable:	Desarrollador		
Colaborador:	Usuario		
Descripción:	• En función a la historia de usuario ingresar al sistema SafePoint se debe ingresar un usuario y contraseña el cual permitirá ingresar al sistema.		
Pasos de ejecución:	• Click en el icono del sistema SafePoint • Ingresar usuario y contraseña • En caso de no tener, indicar al usuario de administración que cree su usuario y contraseña • En caso de olvido el administrador será quien recupere su contraseña o usuario.		

Resultado esperado:	• En las pruebas el usuario se equivocó de clave el cual le alerto de clave incorrecta. • Ingresó correctamente
Resultado Obtenido:	• Se obtiene los resultados esperados

Fuente: Propia.
Elaborado por: El Autor

Tabla 25 Prueba de aceptación PA2

PRUEBA DE ACEPTACIÓN			
Nombre de la prueba:	Creación de Perfiles de Usuarios	**ID:**	**PA2**
Responsable:	Desarrollador		
Colaborador:	Usuario		
Descripción:	• En la historia de usuario creación de perfiles de usuarios, permite crear perfiles de usuarios quien tendrán acceso a diferentes accesos a los menús		
Pasos de ejecución:	• Ingresa al menú Administración • Ingresar a Seguridad • Ingresa a Perfiles • Selecciona los permisos requeridos. • Ingresa los datos solicitados • Guardar, Modificar, Eliminar o Salir.		
Resultado esperado:	• Al guardar despliega un mensaje indicando que campos falta por llenar. • Se guardó exitosamente.		
Resultado Obtenido:	• Se obtiene los resultados esperados		

Fuente: Propia.
Elaborado por: El Autor

Tabla 26 Prueba de aceptación PA3

PRUEBA DE ACEPTACIÓN			
Nombre de la prueba:	Ingresar Artículos	**ID:**	**PA3**
Responsable:	Desarrollador		
Colaborador:	Usuario		
Descripción:	• En función a la historia de usuario ingresar artículo, se permitirá ingresar la información por nombre, detalle y aplicación por cada artículo.		
Pasos de ejecución:	• Ingresar al menú Administración • Ingresar al sub menú Parámetros • Ingresar al sub menú Artículos • Ingresar los campos solicitados • Guardar, Editar, Eliminar artículo		

Resultado esperado:	• No permitió ingresar letras en los campos costo y precio de venta. • Para poder guardar solicita que se llene todos los campos. • Se guardó satisfactoriamente.
Resultado Obtenido:	• Se obtiene los resultados esperados en las pruebas en artículos

Fuente: Propia.
Elaborado por: El Autor

Tabla 27 Prueba de aceptación PA4

PRUEBA DE ACEPTACIÓN			
Nombre de la prueba:	Ingresar Clientes	**ID:**	**PA4**
Responsable:	Desarrollador		
Colaborador:	Usuario		
Descripción:	• En función a la historia de usuario ingresar cliente, se permitirá ingresar la información por tipo de cédula.		
Pasos de ejecución:	• Ingresar al menú Administración • Ingresar al sub menú Parámetros • Ingresar al sub menú Cliente • Ingresar los campos solicitados • Guardar, Editar, Eliminar Cliente		
Resultado esperado:	• No permitió guardar por no tener un email válido. • Para poder guardar solicita que se llene todos los campos. • Se guardó satisfactoriamente.		
Resultado Obtenido:	• Se obtiene los resultados esperados en las pruebas en artículos		

Fuente: Propia.
Elaborado por: El Autor

Tabla 28 Prueba de aceptación PA5

PRUEBA DE ACEPTACIÓN			
Nombre de la prueba:	Ingresar Proveedor	**ID:**	**PA5**
Responsable:	Desarrollador		
Colaborador:	Usuario		
Descripción:	• En función a la historia de usuario ingresar proveedor, se permitirá ingresar la información por tipo de ruc.		
Pasos de ejecución:	• Ingresar al menú Administración • Ingresar al sub menú Parámetros • Ingresar al sub menú Proveedor • Ingresar los campos solicitados • Guardar, Editar, Eliminar Cliente		
Resultado esperado:	• No permitió guardar por no tener un email válido. • Para poder guardar solicita que se llene todos los campos. • Se guardó satisfactoriamente.		
Resultado Obtenido:	• Se obtiene los resultados esperados en las pruebas en artículos		

Fuente: Propia.
Elaborado por: El Autor

Tabla 29 Prueba de aceptación PA6

PRUEBA DE ACEPTACIÓN			
Nombre de la prueba:	Consulta de movimiento de stock	**ID:**	**PA6**
Responsable:	Desarrollador		
Colaborador:	Usuario		
Descripción:	• En función a la historia de consulta de movimiento de stock, se los puede visualizar al ingresar al sistema.		
Pasos de ejecución:	• Ingresar al menú movimientos • Ingresar al sub menú Venta • Seleccionar Cliente • Seleccionar Artículo • Ingresar los campos orden de trabajo y encargado • Guardar o Salir. • Ingresamos a menú movimientos • Ingresamos al sub menú reportes. • Ingresamos al sub menú Ajustes/Stock • Validar si se restaron los stock vendidos		
Resultado esperado:	• Al no ingresar cantidad no se ingresa el producto. • Al no ingresar productos no guarda la venta. • Al no ingresar los campos requeridos indica cuales son los requeridos. • Se guarda la venta • No pudo continuar con la venta por no ingresar cantidad del producto seleccionado. • No se pudo guardar la venta por no llenar los campos faltantes. • Solo puede ingresar números mayores a cero y solo números.		
Resultado Obtenido:	• Se obtiene los resultados esperados exitosamente.		

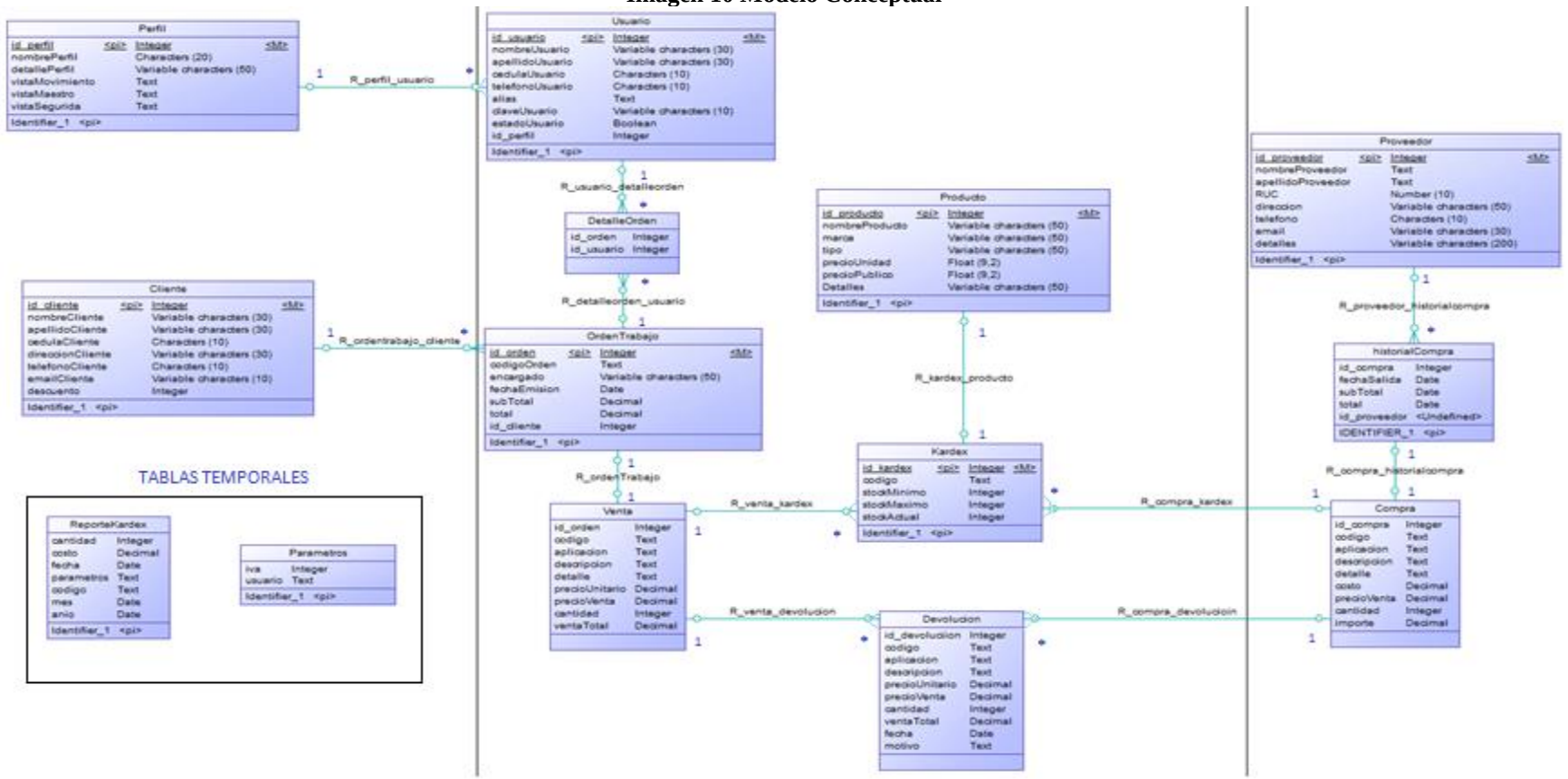
Fuente: Propia.
Elaborado por: El Autor

3.2 Esquema de la Base de Datos

A continuación se describe las clases, atributos y las relaciones que contiene la base de datos.

3.2.1 Modelo Conceptual

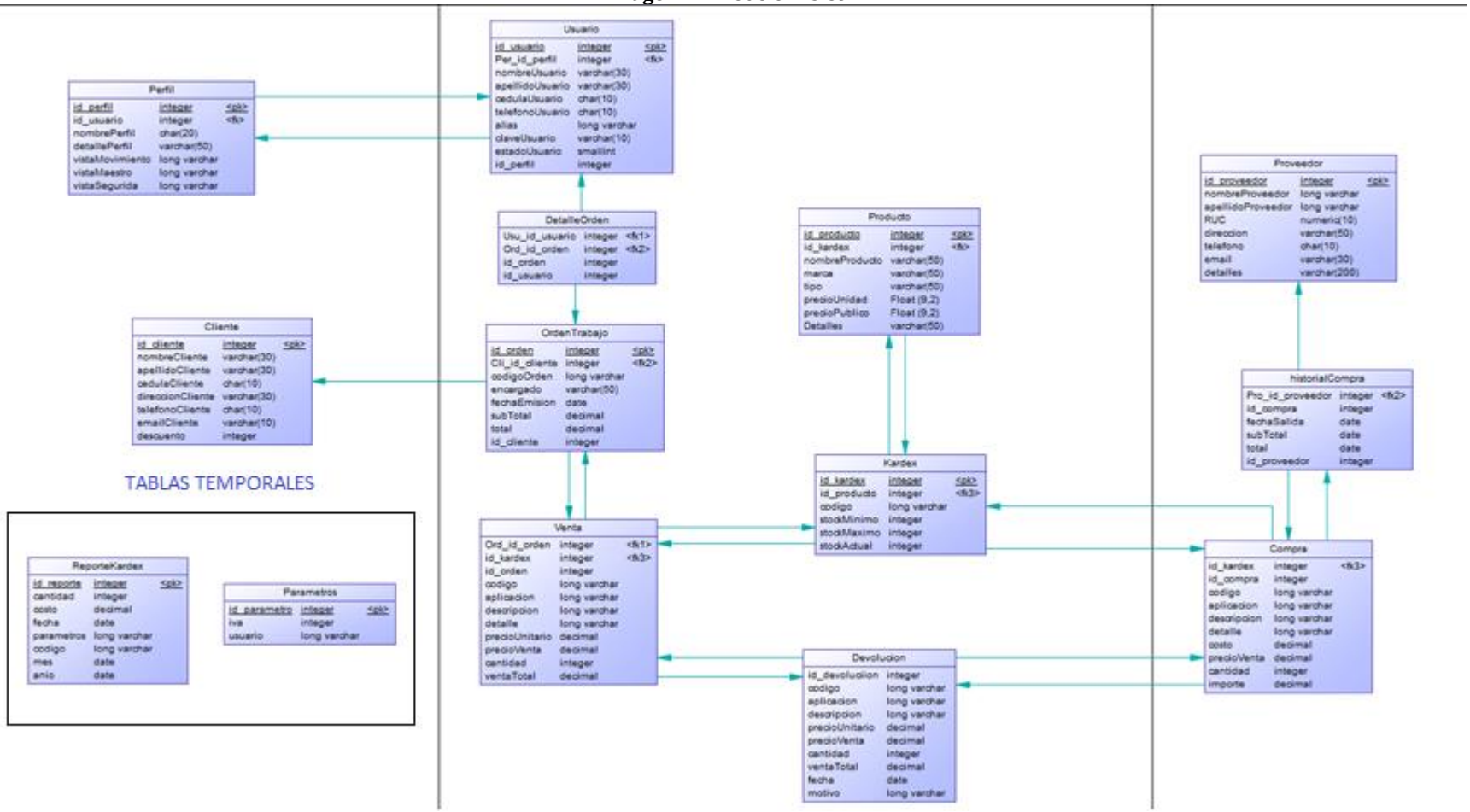
Imagen 10 Modelo Conceptual



Fuente: Propia.
Elaborado por: El Autor

3.2.2 Modelo Físico

Imagen 11 Modelo Físico

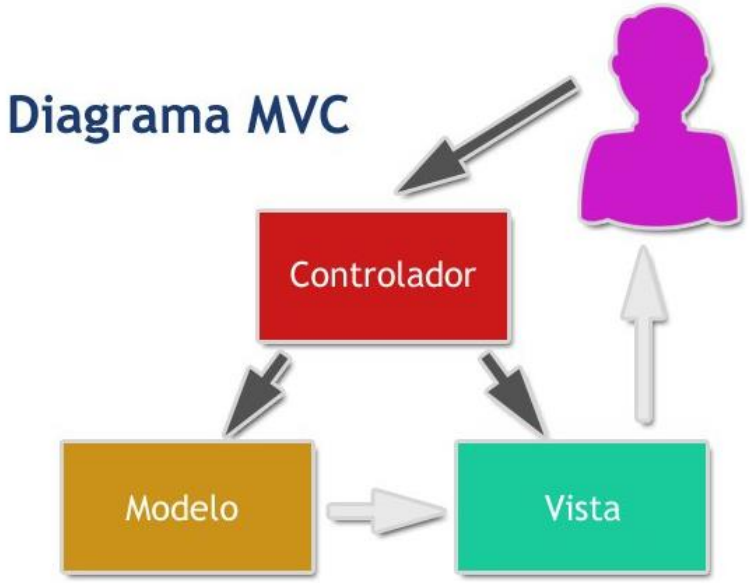


Fuente: Propia.
Elaborado por: El Autor

3.3 Diagrama de la arquitectura del sistema

La arquitectura está basada en el modelo MVC (models, views and controllers), por su facilidad de mantenimiento y de reutilización de código.

Imagen 12 Arquitectura MVC

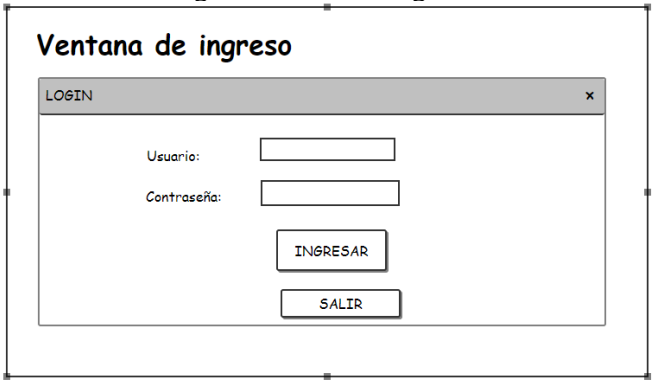


Fuente: (Alvarez, 1999)

3.4 Diseño de interfaces

Las siguientes pantallas detalladas a continuación son las principales empezando por la ventana login el cual consiste en validar si es un usuario autorizado para ingresar al sistema informático, cabe recalcar que la clave se encuentra encriptado en la base de datos siendo así más segura y confiable.

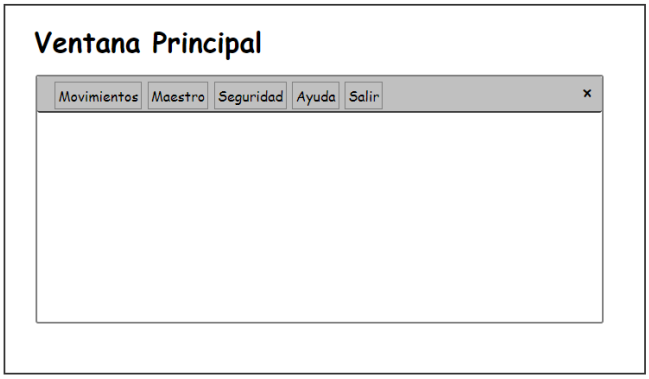
Imagen 13 Venta De Ingreso



Fuente: Propia
Elaborado por: El Autor

La siguiente ventana es la principal el cual consta de 4 menús principales las cuales se tendrá acceso según el perfil del usuario.

Imagen 14 Ventana Principal



Fuente: Propia
Elaborado por: El Autor

Venta Movimientos es el primer submenú de la venta principal el cual consta de submenús (compra, venta, reportes).

Cuando se realiza una venta de artículo se debe ingresar al submenú "Venta", si hace falta repuestos se debe ingresar a "Compra" donde se selecciona el tipo de proveedor y el artículo comprado.

Todo tipo de transacción se guarda en la base de datos, el cual con la opción "Reportes" se puede visualizar lo realizado.

Imagen 15 Ventana Movimiento

Fuente: Propia
Elaborado por: El Autor

Venta Maestro la cual consta de objetos que se van a crear como son: clientes, proveedores, artículos. Cada opción puede también exportar en archivo Excel los datos que existen en la base de datos por cada tipo de opción.

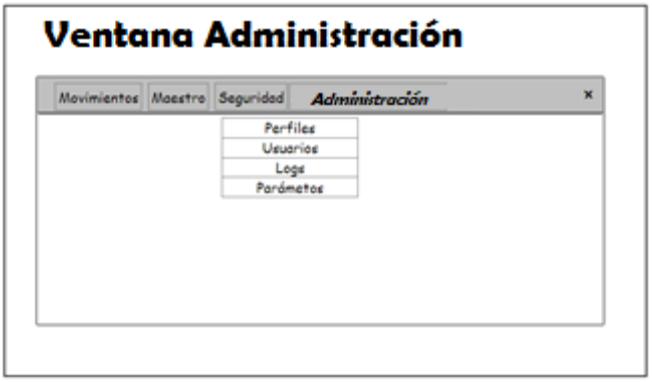
Imagen 16 Ventana Maestro

Fuente: Propia
Elaborado por: El Autor

Ventana seguridad consta sobre accesos, los logs para auditoría y parámetros globales. El cual solo administradores son los encargados de modificar por seguridad de la empresa por un posible fraude.

Aquí se puede cambiar el stock que tiene actualmente la Bodega cada artículo, como también se puede realizar un back up de la base de datos cuando crear necesario el administrador.

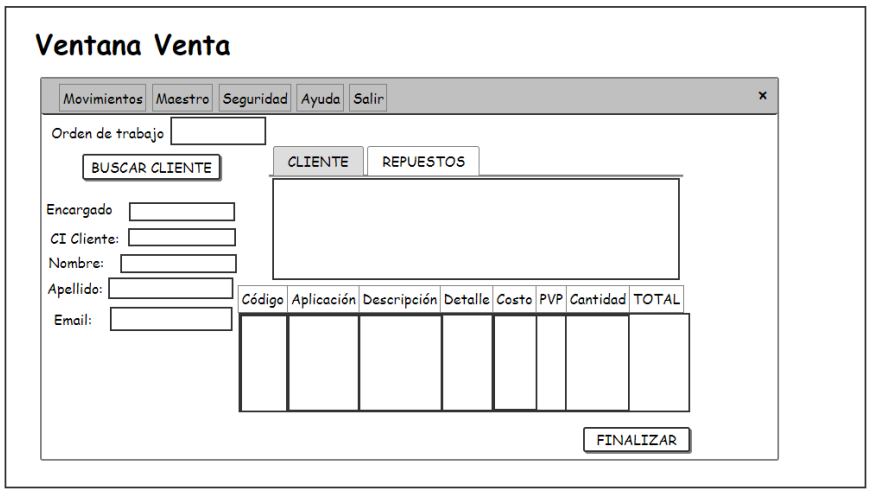
Imagen 17 Ventana Administración



Fuente: Propia
Elaborado por: El Autor

Ventana venta la cual consta de dos sub pantallas donde debe seleccionar el cliente seguido de los repuestos a seleccionar para una venta.

Imagen 18 Ventana Venta



Fuente: Propia
Elaborado por: El Autor

3.5 Estándares de programación utilizados

La metodología utilizada es XP la cual facilita la comprensión del código para cualquier otro desarrollador lo pueda entender no solo el creador del sistema.

3.5.1 Base de datos

Para los estándares de bases de datos se aplicaron:

- Para el nombre de la base de dato se utiliza la primera letra con mayúsculas el resto con minúsculas ejemplo *Proyecto.*
- Para el nombre de las tablas se usa con minúsculas, si tiene dos palabras la segunda empezará minúsculas al igual que el resto ya que PostgreSQL no puede identificar mayúsculas en medio de dos palabras ejemplo *ordentrabajo*.
- Los nombres de los campos también se utiliza con minúsculas, si tiene varias palabras la segunda empezará con la primera letra en mayúscula el resto minúsculas ejemplo *nombreCliente*
- Para las claves primarias se utiliza el nombre de la tabla seguido con _pkey, ejemplo *kardex_pkey*.

- Para las claves foráneas se utiliza fk_ seguido con el campo a relacionar, ejemplo *fk_idCodigo.*
- Para el nombre de las vistas se utiliza el nombre vista_ más el nombre de la tabla1 seguido por el segundo nombre de la tabla2 ejemplo *vista_reporte_venta*.

3.5.2 Código de programación

Los estándares para el código de programación se aplicaron:

- Para el nombre de los paquetes se pone la palabra inicial ***com.*** ,seguido del nombre nuevo del paquete ejemplo *com.panel*
- Para el nombre de las clases la primera letra se la coloca en mayúsculas, el resto en minúsculas ejemplo *Cliente*
- Para el nombre de paneles se ingresa la palabra panel seguido del nuevo nombre la primera letra con mayúscula, ejemplo *panelCatalogo*
- Para el nombre de reportes se coloca la palabra reporte seguido del nuevo nombre y su primera letra será en mayúscula ejemplo *reporteVenta*
- El nombre de las vistas o form el nombre empieza con frm seguido del nuevo nombre con su primera letra con mayúscula ejemplo *frmRepuesto*
- Para los nombres de las variables se usa palabras en minúsculas si tiene más palabras se combina con la primera letra con mayúscula y el resto con minúsculas, ejemplo *columnaBuscar*.
    - Para los diferentes controles en los formularios se usa como referencia la tabla:

Imagen 19 Estándares de programación

Tipo de Control	Prefijo	Ejemplo
JCheckBox	chk	chkProducto
JComboBox	cbo	cboTipoProducto
JButton	btn	btnAceptar
JForm	frm	frmMantenimiento
JPanel	pnl	pnlDatosCliente
JLabel	lbl	lblUsuario
JToolBar	tob	tobInicio
JTextLine	txt	txtNombre
JTextArea	txa	txaDescripcion
JTable	tbl	tblRuta
JMenuBar	mub	mubBarraHerramientas
JMenu	mnu	mnuInicio
JMenuItem	mui	muiAdministrarCliente
JFileChooser	flc	flcBuscarDocumento
JOptionPane	otp	otpMensajeConfirmacion
JFrame	frm	frmIncioSesion

Fuente: (Abanto, 2011)

Tanto para la base de datos como para el código fuente de la aplicación se encuentra identado correctamente.

3.6 Implementación

Debido que no existe anteriormente un software implementado ni una base de datos existente, la fase de implementación se lo realizara con la ayuda del personal de Bodega y el departamento de venta quienes tienen a su poder la información de los repuestos.

3.6.1 Plan de implementación

Tabla 30 Plan de implementación

ACTIVIDAD	REFERENCIA	TIEMPO
Adquirir un equipo informático para la instalación de la base de datos y la aplicación.	Está en proceso de compra por parte de la gerencia e inversionistas.	1 Semana
Instalar JDK 8.0 para aplicaciones Java	Es requerido para el funcionamiento de aplicaciones Java	2 horas
Instalación de Netbeans IDE 8.2	Para poder solventar imprevistos en la empresa	2 horas
Instalación de PgAdmin	Administrador para la revisar motores PostgreSQL	1 hora
Instalación PostgreSQL 10	Motor de base de datos	1 hora
Crear los tipos de perfiles y usuarios para el ingreso al sistema y acceso a sus diferentes componentes.	Esto se los creará con la ayuda del gerente de la empresa CarPoint el cual indicara que empleados tendrán accesos al sistema.	1 hora
Ingresar al sistema informático los diferentes tipos de repuestos existentes en la base de datos mediante la utilización del sistema informático	Con la ayuda del personal de Bodega se ingresará los diferentes artículos que exista en stock	1 semana
Hacer pruebas de funcionalidad ingresando nuevos pedidos y comparando con el stock real de Bodega.	Se hará uso de las hojas de inventario que posee la empresa CarPoint, donde constan todos los repuestos que existen actualmente.	3 días

Validar su veracidad de los reportes que se generan en el sistema	Con la ayuda del encargado de Bodega se validará las existencias.	3 días
Generar reportes con el sistema informático	Con la carpeta kárdex comparar los reportes generados del sistema.	3 días

Fuente: Propia.
Elaborado por: El Autor

3.6.2 Requerimiento de SW y HW

Para el correcto funcionamiento del programa se requiere las siguientes especificaciones:

Tabla 31 Requerimientos de HW/SW

Programa	Versión	Descripción	Requisitos mínimos
NetBeans	IDE 8.2	Es un producto libre y gratuito sin restricciones de uso, sirve para desarrollar con el lenguaje de programación Java.	Microsoft Windows Vista SP1 / Windows 7 Professional: Procesador: Intel Pentium III a 800 MHz o equivalente Memoria: 512 MB Espacio en disco: 750 MB de espacio libre en disco
PgAdmin	4	Es una plataforma de administración de base de datos para PostgreSQL	Windows 2000 y superior, hasta la versión 1.14.3. Desde v1.16.0, se requiere Windows XP / 2003 o posterior, y desde 1.20.0, se requiere Windows Vista / 2008 Procesador: 1GHz o superior Memoria: 256 MB Espacio en disco: 300 MB de espacio libre en disco
PostgreSQL	10	Es un motor de base de datos basado en objetos y es open-source	Windows XP o superior Procesador: 1 GHz o superior Memoria: 1 GB Espacio en disco: 512 MB de espacio libre en disco

Fuente: Propia
Elaborado por: EL Autor

3.6.3 Manual de Usuario

Como su nombre lo indica sirve como un manual de funcionamiento del software para el usuario. El manual de usuarios se encuentra en el Anexo 3.

3.6.4 Manual Técnico

A continuación se detallará los pasos necesarios para un óptimo funcionamiento del software a implementar. Ver anexo 4.

3.6.5 Plan de Capacitación

En la siguiente tabla se especifica el plan de capacitación que se llevará a cabo en la empresa CarPoint

Tabla 32 Plan de Capacitación

N°	ACTIVIDAD	HORAS	PERSONAL
1	**Menú Movimientos**	**9 horas**	
1.1	Sub menú venta	3 horas	Todos los usuarios de la empresa
1.2	Sub menú compra	3 horas	Todos los usuarios de la empresa
1.3	Sub menú devolución	2 horas	Todos los usuarios de la empresa
1.4	Sub menú reportes	1 hora	Todos los usuarios de la empresa
2	**Menú Maestro**	**3 horas**	
2.1	Sub menú clientes	1	Usuarios administradores
2.2	Sub menú proveedor	1	Usuarios administradores
2.3	Sub menú artículos	1	Usuarios administradores
2	**Menú Administración**	**3 horas**	
2.1	Sub menú seguridad	1	Usuarios administradores
2.2	Sub menú parámetros	1	Usuarios administradores
2.3	Sub menú otros	1	Usuarios administradores

Fuente: Propia
Elaborado por: El Autor

4 CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- En general se puede concluir que los objetivos propuestos para este desarrollo de software se cumplieron según lo planeado, con un control automatizado sobre el inventario de Bodega, así como un control eficaz sobre los repuestos que ingresan y egresan de Bodega.
- De acuerdo a lo evidenciado sobre la arquitectura de datos se pudo observar que no existe una buena representación de la realidad del proceso de Bodega, con el software se tiene una arquitectura de datos organizado y sistematizado.
- El proceso a nivel Bodega, gracias a las metodologías aplicadas al proyecto, se propone una nueva estrategia de solución para el control de existencias de Bodega.
- Para la creación del sistema de Bodega y gracias a la colaboración del personal de la empresa, se implementa correctamente el sistema de Bodega satisfactoriamente.

4.2 Recomendaciones

- Se recomienda para el desarrollo y uso de la aplicación se tenga una maquina específicamente para Bodega, por motivos que pueden ocupar muchos recursos de memoria.
- Es aconsejable tener un software que también maneje la parte contable de la empresa CarPoint, con esto facilitaría las tareas para las personas del área de contabilidad.
- Es necesario que la empresa acceda a los servicios que ofrece el internet para que pueda experimentar las distintas alternativas que ofrece la tecnología para generar mayor productividad y a bajos costos.
- Finalmente cada día mejora la tecnología para desarrollar software, por lo que se recomienda invertir en sistemas informáticos, con la finalidad de fortalecer el sistema de inventario como también sus procesos dentro de su empresa.

5 BIBLIOGRAFÍA

Picón, M. V. (1998). *El kárdex como herramienta contable.* Lima.

Abanto, L. (2011). *Documento de Estándares de Programación.* Madrid: Sistema Thor Store.

Abascal, M. M., Pérez, N. N., & Góngora, H. C. (2010). *Propuesta de migración de soluciones de base de datos que utilizan Oracle hacia postgresql.* Cuba: Editorial Universitaria.

Albaladejo, X. (2015, 09). *proyectos agiles.org*. Retrieved from https://proyectosagiles.org/

Albalejo, X. (2015, 09). *proyecto agiles*. Retrieved from https://scrumenespanol.files.wordpress.com/2015/09/diagrama-proceso-scrum.gif?w=553&h=414

Alvarez, M. A. (1999, Diciembre). *DesarrolloWeb.com*. Retrieved from DesarrolloWeb.com: https://desarrolloweb.com

Arley, R. C. (2011). *El Software libre.* Costa Rica: Red Universidad Nacional de Costa Rica.

AutoSporte. (2017, 06 17). *AutoSporte*. Retrieved 12 2017, 08, from AutoSporte: http://www.autosoporte.com/blog-automotriz/item/298-que-es-un-taller-mecanico

Ballou, R. (2004). *GestioPolis.com.* Retrieved 12 2017, 08, from GestioPolis.com: https://www.gestiopolis.com/que-es-inventario-tipos-utilidad-contabilizacion-y-valuacion/

Bravent IT consulting company. (2015, 10). *Bravent*. Retrieved from http://info.bravent.net/blog/scrum-metodologias-agiles-beneficios-aportan-al-desarrollo-del-software

Finanzas y contabilidad Copyright © 2018. (2016, 07). *FINANZAS Y CONTABILIDAD.* Retrieved from https://finanzascontabilidad.com/valoracion-existencias/

Finanzas y contabilidad Copyright. (2016, 07). *FINANZAS Y CONTABILIDAD.* Retrieved from https://finanzascontabilidad.com/

Flamarique, S. (2018). *Gestión de existencias en el almacén.* Barcelona: Marge Books.

García de Jalón, J., Rodríguez, J. I., Mingo, I., Imaz, A., Brazález, A., Larzabal, A., . . . García, J. (2000). *Aprenda Java como si estuviera en primero.* San Sebastían .

Gonzalo, S. V. (2009). *Contabilidad Administrativa.* Bogotá: Eco.

Joskowicz, I. J. (2008). *Reglas y Prácticas en eXtreme Programming.* Vigo , España: Universidad.

Martínez, Á. M., & Celis, F. M. (2015). *Contabilidad General con enfoque NIIF para las pymes.* Bogotá: Ecoe.

Oracle. (2014, 08 08). *¿Qué es la tecnología Java y para qué la necesito?* . Retrieved 12 08, 2017, from ¿Qué es la tecnología Java y para qué la necesito? : view-source:https://www.java.com/es/download/faq/whatis_java.xml

PostgreSQL. (2015). *Qué es PostgreSQL y cuáles son sus ventajas.* Retrieved 12 08, 2017, from Platzi: https://platzi.com/blog/que-es-postgresql/

Quintana, G., Marqués, M., & Aliaga, J. (2010). *Aprende SQL.* Castelló de la Plana: Universitat Jaume I. Servei de Comunicació i Publicacions.

Rodriguez Gonzáles, M. E. (2013). *Gestíon de datos.* Catalán: Editorial UOC.

Rubio Ferrer , J., & Villarroel Valdemoro, S. (2012). *Gestión y pedido de stock.* Ministerio de Educación de España: Valencia.

Sanchez Allende, J., Fernandez Toribio, G., & Moreno Díaz, P. (2005). *Programación en Java 2.* Madrid: McGraw-Hill España.

SOFTENG. (2011, 03 01). *SOFTENG.* Retrieved from https://softeng.blob.core.windows.net/softengpublish/invar/7933ab40-b512-4acc-8b49-e37ebedddd45

SRI. (2016, 06). *SOLICITUD PARA AUTORIZACION DE AUTOIMPRESORES DE COMPROBANTES DE VENTA Y RETENCIÓN.* Ecuador.

Stallman, R. M. (2004). *Software libre para una sociedad libre.* Versión 1.0.

Tejero, J. J. (2008). *Almacenes Análisis,diseño y organización.* Madrid: ESIC.

Yolanda, M. V. (2010). *Industría Automotriz y Automatización.* Tlalpan: Centro de Investigación y Estudios Superiores México.

ANEXOS

Anexo 1: Encuesta realizada al personas de la empresa CarPoint

¿La Bodega es una parte esencial dentro de su empresa?

¿Cree usted que es favorable implementar un sistema informático efectivo, para un adecuado manejo de los repuestos en Bodega?

¿Cree usted que es eficiente y confiable, la manera que es llevada actualmente el Control de Repuestos en Bodega?

¿Estaría dispuesto a realizar un cambio total sobre el manejo del control de Bodega?

¿Estaría dispuesto a invertir en un Sistema Informático para cambiar procesos en su Empresa?

¿Le gustaría que todo empleado tenga un control de acceso para el manejo de Bodega?

¿Cree usted que un buen manejo de control Informático en Bodega, agilitaría el trabajo de cada empleado de su Empresa?

¿Considera que al poseer un Sistema Informático de Bodega, abría un control más efectivo de equipos, herramientas, etc. necesarias para su actividad?

¿La información de Bodega que llega al departamento de contabilidad es clara y concisa? SI o NO ¿Por qué?

¿El manejo de Bodega es llevada con algún procedimiento o norma?

¿Tienen respaldo de información sobre los repuestos y stock de Bodega?

Anexo 2

	PERSONAL DE CARPOINT							
	Administración		**Profesionales**			**Empleados**		
Preguntas	**Inversionista 1**	**Inversionista 2**	**Jefe de taller**	**Mecánico 1**	**Mecánico 2**	**Ayudante 1**	**Ayudante 2**	**Bodeguero**
¿La Bodega es una parte esencial dentro de la empresa?	Si	Si	Si	Si	Si	Talvez	Si	Si
¿Cree usted que es favorable implementar un sistema informático efectivo, para un adecuado manejo de los repuestos en Bodega?	Si	Si	Si	Talvez	Si	Talvez	Si	Talvez
¿Cree usted que es eficiente y confiable, la manera que es llevada actualmente el Control de Repuestos en Bodega?	Talvez	No	Talvez	Talvez	Talvez	Talvez	No	Si
¿Estaría dispuesto a realizar un cambio total sobre el manejo del control de Bodega?	No	Si	Si	Talvez	Talvez	Talvez	Talvez	Talvez
¿Estaría dispuesto a invertir en un Sistema Informático para cambiar procesos en su Empresa?	Si	Si	Si	Si	Talvez	Si	Talvez	Talvez
¿Le gustaría que todo empleado tenga un control de acceso para el manejo de Bodega?	Talvez	Si	Si	Talvez	Talvez	Talvez	Talvez	Talvez
¿Cree usted que un buen manejo de control Informático en Bodega, agilitaría el trabajo de cada empleado de su Empresa?	Si	Si	Si	Si	Si	Talvez	Talvez	Talvez
¿Considera que al poseer un Sistema Informático de Bodega, abría un control más efectivo de repuestos, equipos, herramientas, etc. necesarias para su actividad?	Si	Si	Si	Si	Si	Si	Si	Si
¿La información de Bodega que llega al departamento de contabilidad es clara y concisa?	No	Talvez	No	No	Talvez	No	No	No
¿El manejo de Bodega es llevada con algún procedimiento o norma?	No	Talvez	Talvez	No	Talvez	No	Talvez	Si
¿Tienen respaldo de información sobre los repuestos y stock de Bodega?	No	No	No	No	No	No	No	No

Anexo 3

UNIVERSIDAD TECNOLÓGICA ISRAEL

MANUAL DE USUARIO

SAFEPOINT

SISTEMA DE INFORMÁTICO PARA LA GESTIÓN DE INVENTARIO PARA LA EMPRESA CARPOINT

RESPONSABLE:
EDISON MARTIN MOLINA SORIA

VERSIÓN 1.0

FECHA DE ELABORACIÓN
16 de Agosto del 2018

TABLA DE CONTENIDO

INTRODUCCIÓN.....2

1 Requerimientos Técnicos de Hardware y Software 2

2 Ingreso al Sistema.....2

3 Ingreso A La Pantalla Principal 3

3.1 Movimiento.....4

3.2 Maestros 6

3.3 Administración 7

4 Glosario.....9

INTRODUCCIÓN

El siguiente documento tiene como finalidad explicar el funcionamiento del software **"SafePoint"** para que la aplicación funcione de manera óptima para los diferentes tipos de usuarios.

SafePoint es el nombre dado al software diseñado para la empresa CarPoint que permite manejar de manera óptima el inventario de Bodega, el usuario al ingresar al sistema tendrá que identificarse con un alias y una contraseña.

El software le permitirá interactuar con el usuario permitiendo ingresar un pedido, crear clientes, crear usuarios, ingresar nuevos artículos y obtener reportes actualizados.

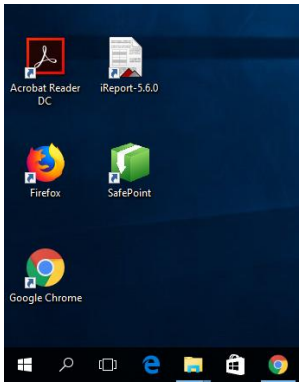
1 Requerimientos Técnicos de Hardware y Software

A continuación se debe tomar en cuenta los siguientes requerimientos mínimos para hardware y software:

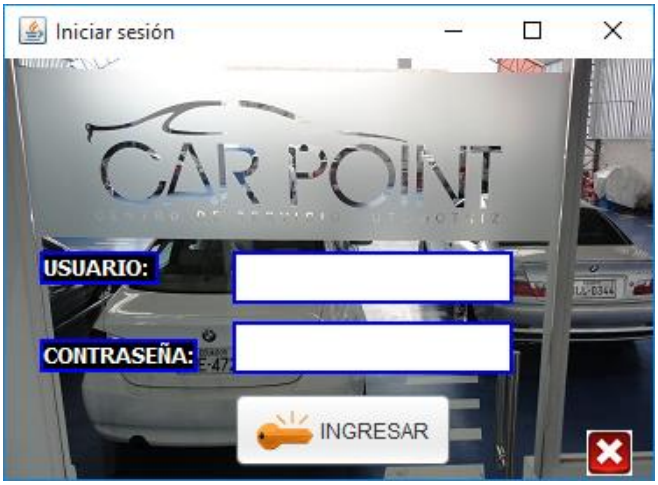
- Windows XP o superior
- Procesador de 1 GHz o superior
- Memoria RAM de 2 GB o superior
- 20 GB de espacio libre en su disco duro como mínimo

2 Ingreso al Sistema

En escritorio dar doble Click en el icono **SafePoint** para ejecutar la aplicación.



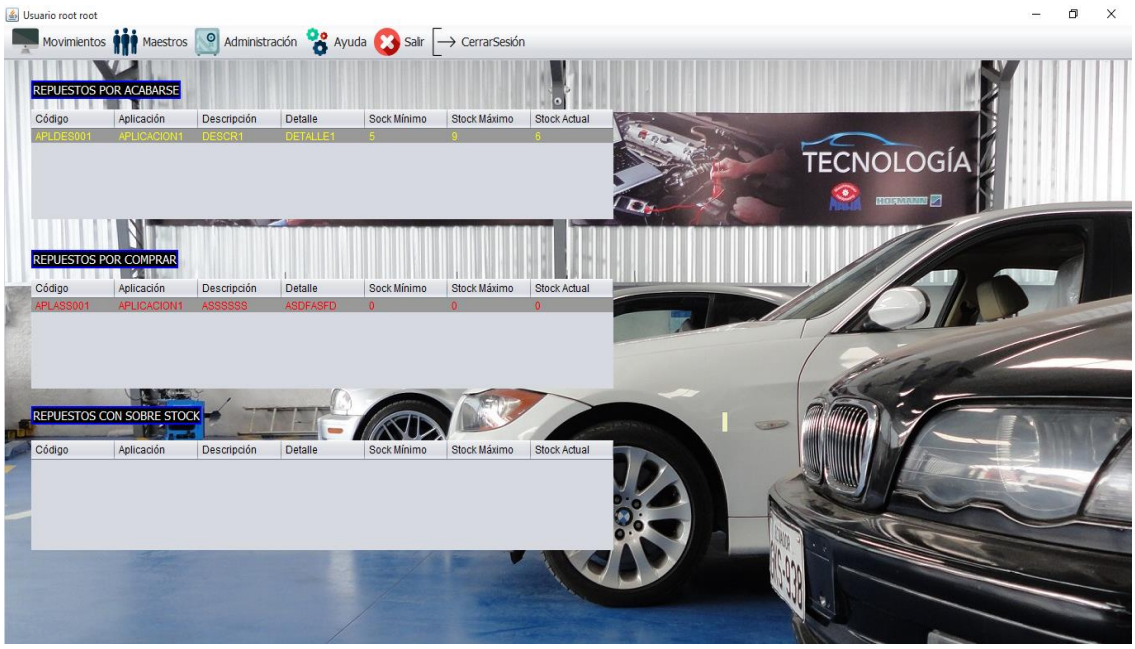
Aparecerá la pantalla de login, el cual el usuario tiene que ingresar el alias y contraseña.



3 Ingreso A La Pantalla Principal

En la pantalla principal como información aparecen 3 tablas las cuales indica:

1. Repuestos por acabarse.- Indica los repuestos que se encuentra en el límite de stock
2. Repuestos por comprar.- Indica los repuestos que están sin stock en Bodega
3. Repuestos con sobre stock.- Indica los repuestos que están excedidos del límite.



La pantalla principal en la barra de menús consta de 6 partes

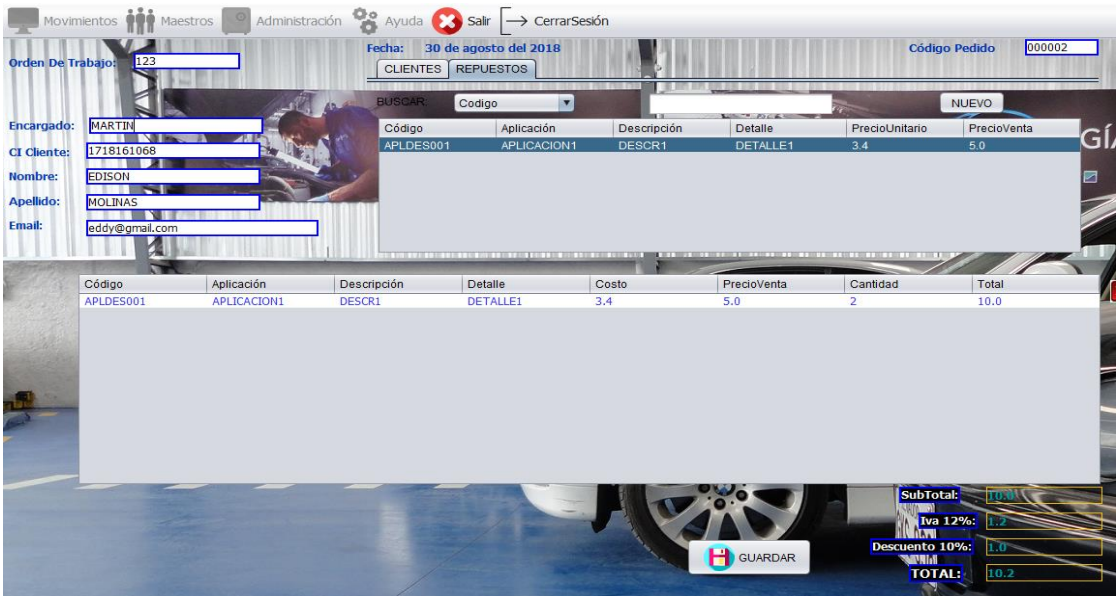
1. Movimientos
2. Maestros
3. Administración
4. Ayuda
5. Salir
6. Cerrar sesión

3.1 Movimiento

Este menú consta de 4 sub menús:

3.1.1 Ventas

En esté sub menú se podrá realizar las ventas que se van a producir en Bodega.



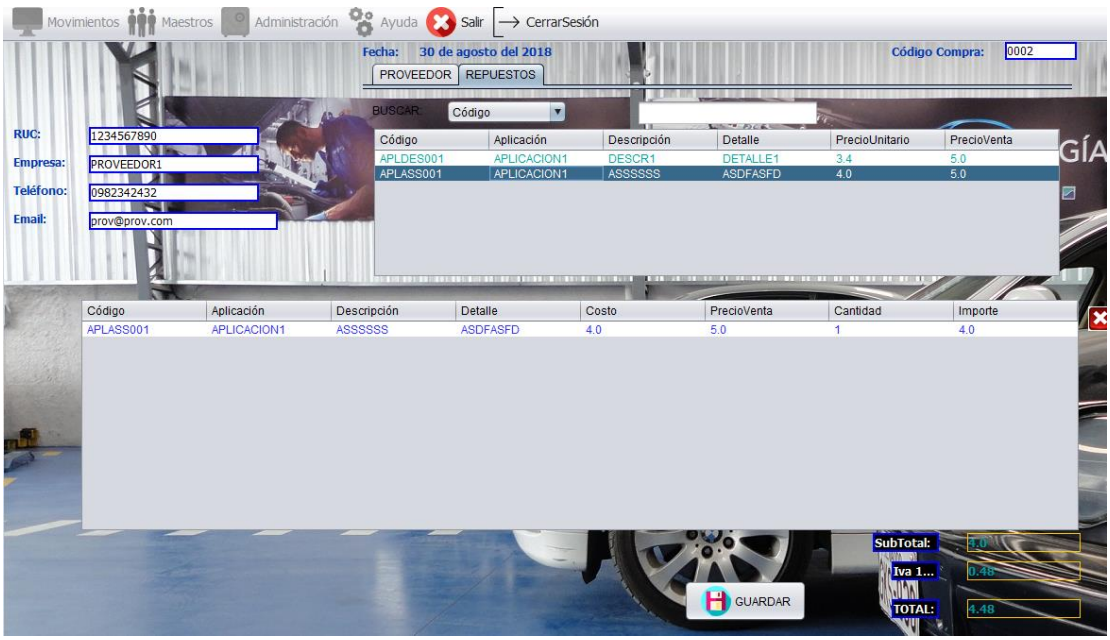
Para realizar una venta se debe ingresar el número de orden que provee la hoja de trabajo, seguido debe ingresar el nombre del Encargado, quien es el que solicita los repuestos.

Click en el botón “Buscar Cliente” el cual es el que carga los datos en una tabla de clientes, el cual provee todo los que existen.

Al seleccionar el cliente automáticamente despliega el otro panel con todos los repuestos que existen en Bodega, el cual deberá seleccionar los que se van a vender el cual se agregaran en el panel de venta, automáticamente le indica el valor total con IVA incluido y descuentos por empleado.

3.1.2 Compra

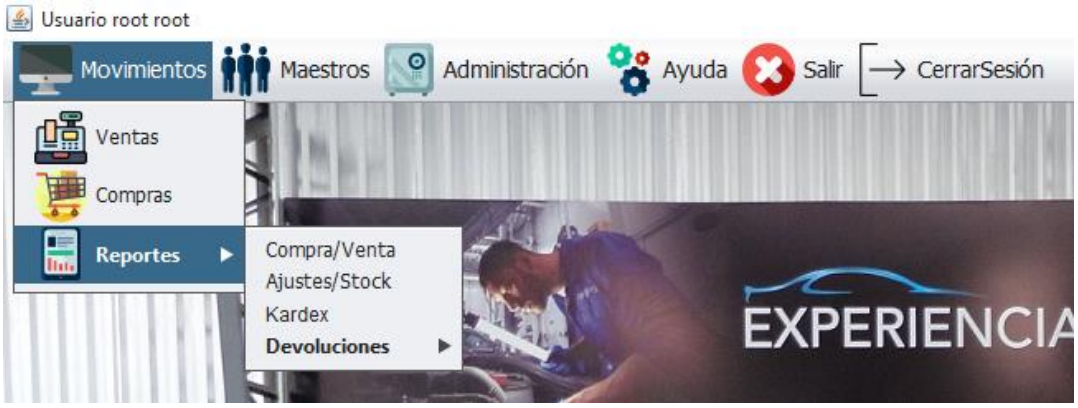
En la siguiente venta se realiza las compras que por lo general son entregadas por proveedores.



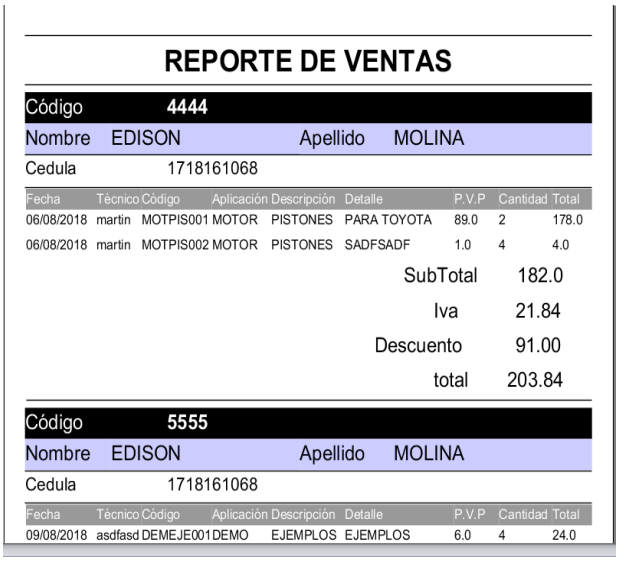
Para realizar una compra se debe seleccionar un proveedor el cual se despliega al dar Click en el botón “Buscar Proveedor”, una vez seleccionado escoger los repuestos que se compraron, una vez ingresado lo requerido se debe guardar los cambios.

3.1.3 Reportes

En esta opción se puede generar reportes de Compra/Venta, Ajustes/Stock, Kárdex y Devoluciones



Para cada tipo de reporte se debe seleccionar la fecha de inicio y fin



REPORTE DE VENTAS

Código	4444		
Nombre	EDISON	Apellido	MOLINA
Cedula	1718161068		

Fecha	Técnico	Código	Aplicación	Descripción	Detalle	P.V.P	Cantidad	Total
06/08/2018	martin	MOTPIS001	MOTOR	PISTONES	PARA TOYOTA	89.0	2	178.0
06/08/2018	martin	MOTPIS002	MOTOR	PISTONES	SADFSADF	1.0	4	4.0

SubTotal	182.0
Iva	21.84
Descuento	91.00
total	203.84

Código	5555		
Nombre	EDISON	Apellido	MOLINA
Cedula	1718161068		

Fecha	Técnico	Código	Aplicación	Descripción	Detalle	P.V.P	Cantidad	Total
09/08/2018	asdfasd	DEMEJE001	DEMO	EJEMPLOS	EJEMPLOS	6.0	4	24.0

3.2 Maestros

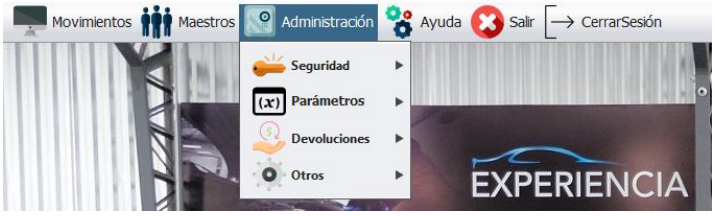
Este menú consta de 3 sub menús Clientes, Proveedores y Artículos.

En estas opciones se crean, eliminar o modifican clientes, proveedores y artículos.

Fuente: Propia
Elaborado por: El Autor

3.3 Administración

Estos submenús que constan de Seguridad, Parámetros, Devoluciones y Otros.



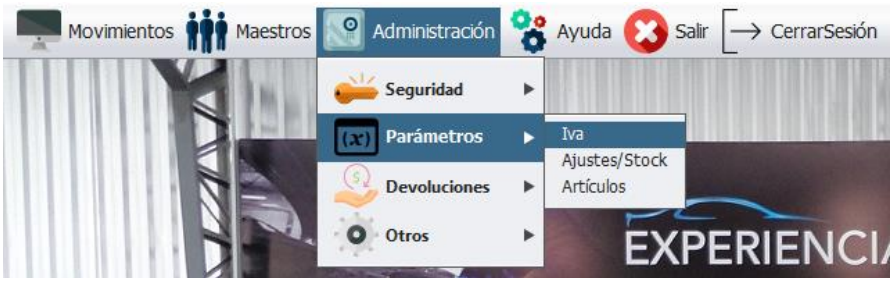
Fuente: Propia
Elaborado por: El Autor

En Seguridad se encuentra los submenús:

1. Perfil donde son los encargados de crear perfiles y cada perfil consta de usuario.
2. Usuario cada uno está atado a su perfil el cual tendrá acceso a los diferentes menús del programa.

Fuente: Propia
Elaborado por: El Autor

En Parámetros sirve para cambiar variables globales como por ejemplo el IVA, cliente, parámetros, proveedores y artículos.



En Devoluciones solo el administrador podrá realizarlas por motivos de un posible fraude, en la cual puede realizar una devolución parcial o total de una venta o compra.

En Otros encontramos:

1. LogsAuditoría aquí encontrará historial de los diferentes en un archivo .txt, el cual almacena los diferentes eventos que tiene el programa para posterior revisión.
2. Backup está opción permite realizar un back up de la base de datos cuando el usuario lo requiera.

Para soporte de programación contactarse a los siguientes contactos:

Edison Molina
Email: eddy196molina@gmail.com
Teléfono: 0983-005366

4 Glosario

- **Backup** Respaldo de la base de datos donde contiene toda la información ingresada al sistema informático.
- **LogsAuditoría** Son eventos que se realiza dentro del sistema, por cada una de ellas guarda en un archivo plano llamado log's.
- **IVA** Impuesto de valor agregado.
- **Kárdex** Documento donde se registra el control de entrada y salida de cada artículo.

Anexo 4

UNIVERSIDAD TECNOLÓGICA ISRAEL

MANUAL TÉCNICO

SAFEPOINT

SISTEMA DE INFORMÁTICO PARA LA GESTIÓN DE INVENTARIO PARA LA EMPRESA CARPOINT

RESPONSABLE:
EDISON MARTIN MOLINA SORIA

VERSIÓN 1.0

FECHA DE ELABORACIÓN
16 de Agosto del 2018

TABLA DE CONTENIDO

1 Objetivo 2

2 Definiciones 2

3 Integrantes 2

4 Requerimientos técnicos de Hardware y Software 2

5 Requisitos 2

6 Configuración 3

6.1 JDK Java 3

6.2 PgAdmin 4 3

6.3 PostgreSQL 4

6.4 NetBeans IDE 8.2 5

7 Arquitectura 7

7.1 Clases y Métodos 7

7.2 Base de datos 13

8 Glosario 15

1 Objetivo

El objetivo del siguiente documento tiene como finalidad guiar en la instalación de los programas requeridos para un óptimo funcionamiento del sistema SafePoint.

2 Definiciones

SafePoint es un sistema informático de escritorio cuya finalidad es la de llevar un correcto manejo de inventario de repuestos en Bodega de la empresa CarPoint.

3 Integrantes

Sr. Edison Martin Molina Soria, Analista IT

4 Requerimientos técnicos de Hardware y Software del servidor

A continuación se debe tomar en cuenta los siguientes requerimientos mínimos para hardware y software:

- Windows XP o superior
- Procesador de 1 GHz o superior
- Memoria RAM de 2 GB o superior
- 20 GB de espacio libre en su disco duro como mínimo

5 Requisitos

Se necesita descargar e instalar los siguientes programas

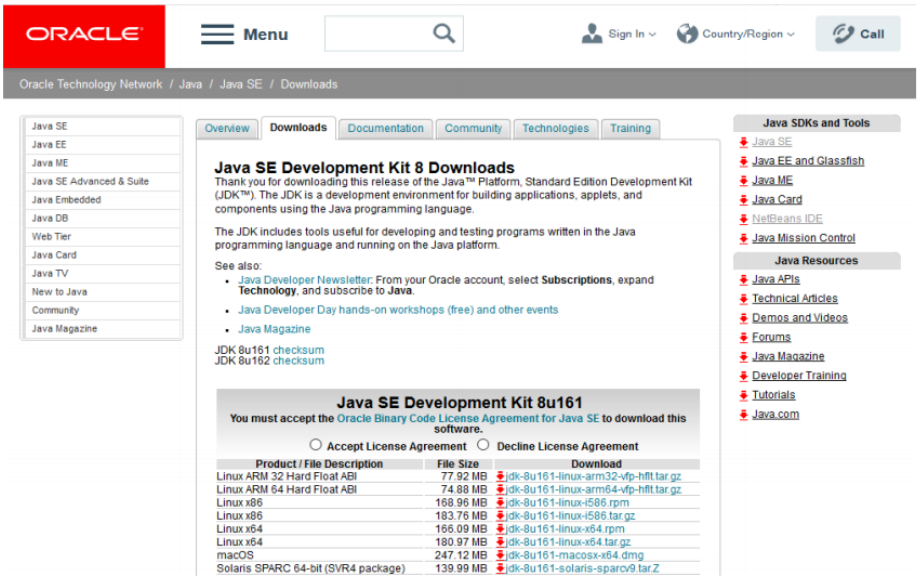
- JDK 8.2
- PgAdmin4 agente de base de datos para el motor PostgreSQL.
- PostgreSQL l0 motor de base de datos para el servidor principal.
- Netbeans IDE 8.2 para el servidor principal.

6 Configuración

6.1 JDK Java

Descargar el JDK 8.2 desde la página oficial de Java

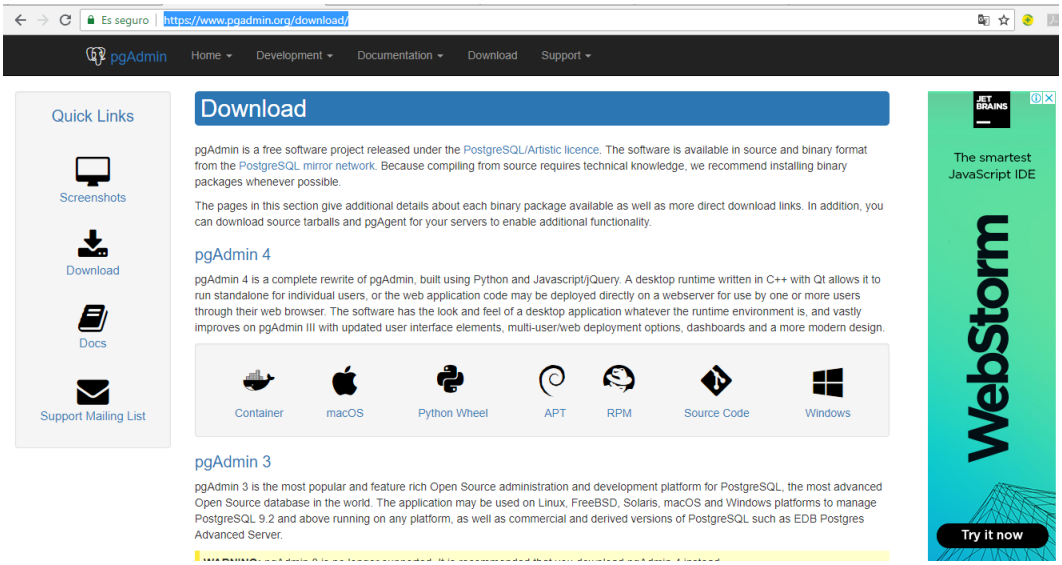
http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html



6.2 PgAdmin 4

Lo puede descargar de la página oficial del siguiente link:

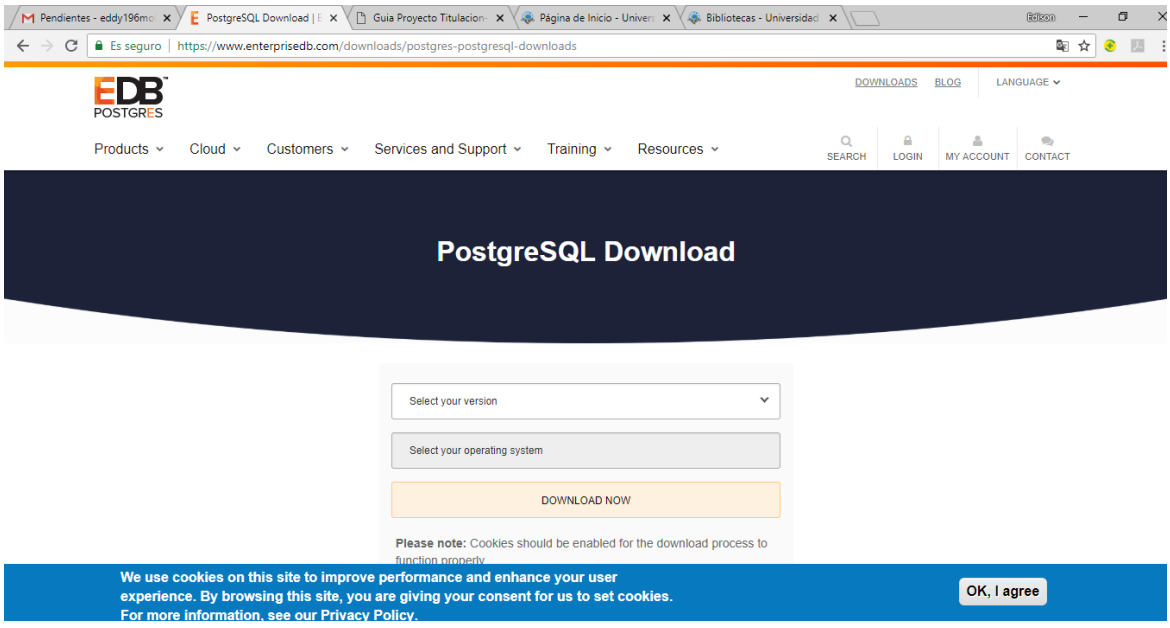
https://www.pgadmin.org/download/



6.3 PostgreSQL

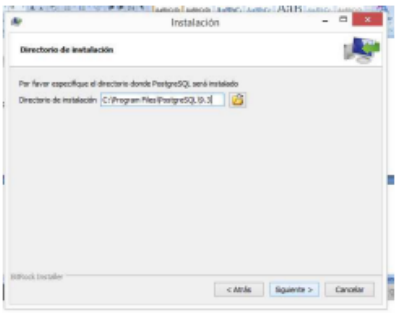
Se lo puede descargar del siguiente link:

https://www.postgresql.org/download/windows/



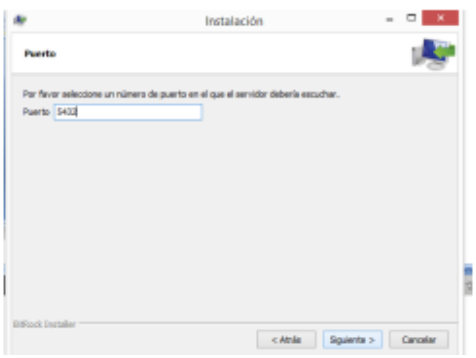
Al instalar el motor PostgreSQL colocar las siguientes variantes:

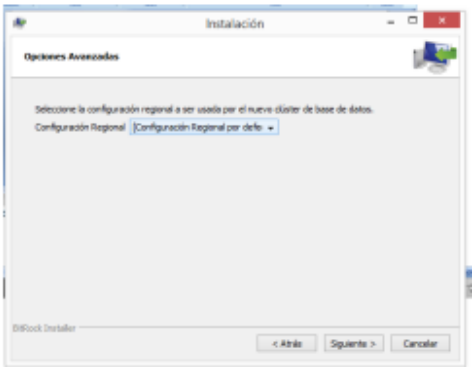
No indicara el directorio donde se va a instalar el PostgreSQL, lo dejamos por defecto y continuamos.



Pondremos una clave en este caso colocaremos “carpoint”, está servirá para posteriormente conectarse a PostgreSQL.

Por seguridad se debe cambiar el puerto por defecto (5432), en este caso colocamos el puerto 5455, y por defecto continuamos seguimos con la instalación.



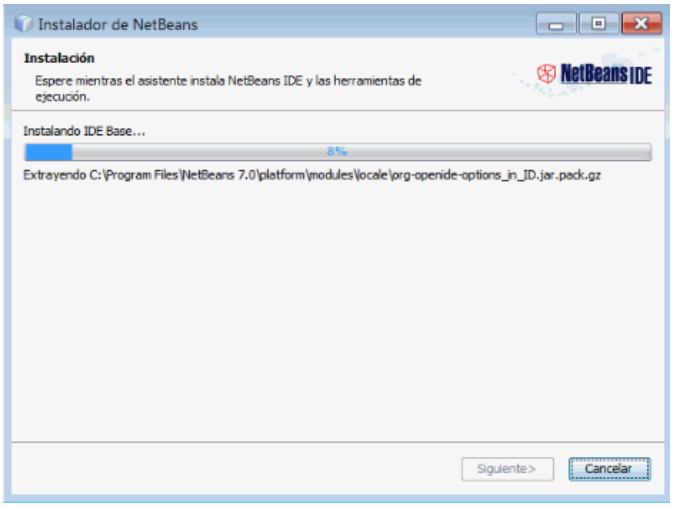


6.4 **NetBeans IDE 8.2**

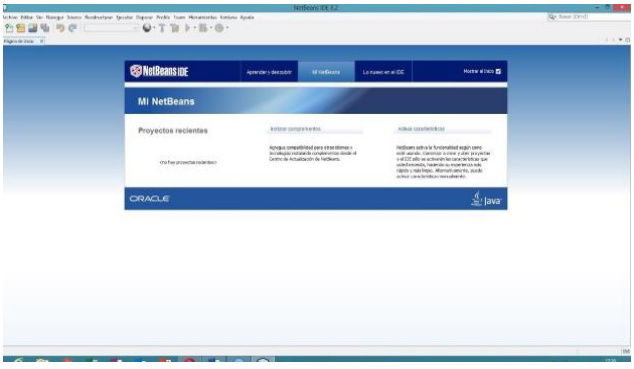
Para NetBeans se debe descargar del siguiente link:

https://netbeans.org/downloads/start.html?platform=windows&lang=en&option=all

Seguimos por defecto, Click en Next hasta que empiece a instalar:



Una vez finalizada la instalación aparecerá el programa NetBeans IDE 8.2

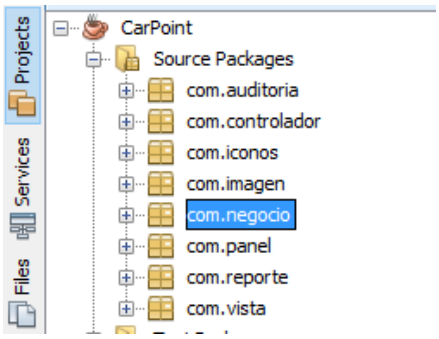


7 Arquitectura

El sistema informático está compuesto de tres módulos principales, uno es la de generar ventas, compras de artículos y generación de reportes.

Los reportes están realizado en “ireport”, es una librería que sirve para diseñar reportes en formato Excel, Word o PDF.

El sistema está compuesto por paquetes en los cuales contiene las clases, paneles, imágenes, iconos, auditoria, controladores, reportes y vistas.



7.1 Clases y Métodos

Las clases principales del programa **“SafePoint”** son:

1. **Conexión:** Es la encargada de establecer conexión con la base de datos mediante la librería postgresql-42.2.4.jar

```
public class conexion {
   static String bd = "Proyecto";
   static String login = "postgres";
   static String url = "jdbc:postgresql://localhost:5454/Proyecto";

   Connection conn = null;

   public conexion(){
      try{
        Class.forName("org.postgresql.Driver");
        conn = DriverManager.getConnection(url,login,password);
        if (conn!=null){
           System.out.println("Conección a base de datos "+bd+" OK");
        }
      }catch(SQLException e){
         JOptionPane.showMessageDialog(null,e+" Revisar conexion a base de datos", "SQLException", JOptionPane.ERROR_MESSAGE);
      }catch(ClassNotFoundException e){
         JOptionPane.showMessageDialog(null,e+" Revisar conexion POSTGRESQL", "ClassNotFoundException", JOptionPane.ERROR_MESSAGE);
      }
   }

   public Connection getConnection(){
      return conn;
   }

   public void desconectar(){
      conn = null;
   }
}
```

2. **Usuario:** Está clase es la encargada de crear, modificar, eliminar los usuarios que podrán interactuar con el sistema informático.

```java
/**
 *
 * @author Edison Molina
 */
public class Usuario {
    private String nombreUsuario,apellidoUsuario,cedulaUsuario,telefonoUsuario,alias,clave;
    private int id_usuario,estadoUsuario,id_perfil;

    conexion cone = new  conexion();
    //Constructor vacio
    public Usuario() {
    }

    //Constructor con datos
    public Usuario(String nombreUsuario, String apellidoUsuario, String cedulaUsuario, String telefonoUsuario, String alias, String clave, int id_usuario,
        this.nombreUsuario = nombreUsuario;
        this.apellidoUsuario = apellidoUsuario;
        this.cedulaUsuario = cedulaUsuario;
        this.telefonoUsuario = telefonoUsuario;
        this.alias = alias;
        this.clave = clave;
        this.id_usuario = id_usuario;
        this.estadoUsuario = estadoUsuario;
        this.id_perfil = id_perfil;
    }

    public String getNombreUsuario() {
        return nombreUsuario;
    }

    public void setNombreUsuario(String nombreUsuario) {
        this.nombreUsuario = nombreUsuario;
    }

    public String getApellidoUsuario() {
        return apellidoUsuario;
    }

    public void setApellidoUsuario(String apellidoUsuario) {
        this.apellidoUsuario = apellidoUsuario;
    }

    public void setClave(String clave) {
        this.clave = clave;
    }

    public int getId_usuario() {
        return id_usuario;
    }

    public void setId_usuario(int id_usuario) {
        this.id_usuario = id_usuario;
    }

    public int getEstadoUsuario() {
        return estadoUsuario;
    }

    public void setEstadoUsuario(int estadoUsuario) {
        this.estadoUsuario = estadoUsuario;
    }

    public int getId_perfil() {
        return id_perfil;
    }

    public void setId_perfil(int id_perfil) {
        this.id_perfil = id_perfil;
    }

    public String getCedulaUsuario() {
        return cedulaUsuario;
    }

    public void setCedulaUsuario(String cedulaUsuario) {
        this.cedulaUsuario = cedulaUsuario;
    }

    public String getTelefonoUsuario() {
        return telefonoUsuario;
    }

    public void setTelefonoUsuario(String telefonoUsuario) {
        this.telefonoUsuario = telefonoUsuario;
    }

    public String getAlias() {
        return alias;
    }

    public void setAlias(String alias) {
        this.alias = alias;
    }

    public String getClave() {
        return clave;
    }
}
```

```java
//Muestra los perfiles en una variable ComboBox con los datos obteniendo de la base de datos perfiles
public DefaultComboBoxModel mostrarPerfil() throws SQLException{
    DefaultComboBoxModel lm = new DefaultComboBoxModel();
    Statement Sentencias = null;
    Sentencias = cone.getConnection().createStatement();
    ResultSet tabla = Sentencias.executeQuery("select nombreperfil from perfil order by id_perfil");
    while(tabla.next()){
        lm.addElement(tabla.getString(1));
    }
    Sentencias.close();
    return lm;
}
//Muestra los usuarios una variable List con los datos obteniendo de la base de datos usuario
public DefaultListModel mostrarUsuario(String perfil) throws SQLException{
    DefaultListModel lm = new DefaultListModel();
    Statement Sentencias = null;
    Sentencias = cone.getConnection().createStatement();
    ResultSet tabla = Sentencias.executeQuery("select nombreusuario,apellidousuario from usuario "
            + "where id_perfil=(select id_perfil from perfil where nombreperfil='"+perfil+"');");
    while(tabla.next()){
        lm.addElement(tabla.getString("nombreusuario")+" "+tabla.getString("apellidousuario"));
    }
    Sentencias.close();
    return lm;
}
```

```java
public void consultaUsuario(String tipo,String perfil)throws SQLException{
    //Divido el String tipo para nombre y apellido
    String[] parts = tipo.split(" ");//Estó une nombre y apellido con un espacio en el medio
    String part1 = parts[0];
    String part2 = parts[1];
    Statement Sentencias = null;
    Sentencias = cone.getConnection().createStatement();
    ResultSet tabla = Sentencias.executeQuery("select id_usuario,cedulausuario,telefonousuario,alias,estadousuario,claveusuario from usuario "
            + "where id_perfil=(select id_perfil from perfil where nombreperfil='"+perfil+"');");
    while(tabla.next()){
        nombreUsuario=part1;
        apellidoUsuario=part2;
        id_usuario=Integer.parseInt(tabla.getString(1));
        cedulaUsuario=tabla.getString(2);
        telefonoUsuario=tabla.getString(3);
        alias=tabla.getString(4);
        estadoUsuario=Integer.parseInt(tabla.getString(5));
        clave=tabla.getString(6);

    }
    Sentencias.close();
}
```

```java
public void actualizarUsuario()throws SQLException{
    Statement Sentencias = null;
    Sentencias = cone.getConnection().createStatement();
    String tabla = "update usuario set nombreusuario='"+trim(getNombreUsuario())+"',apellidousuario='"+trim(getApellidoUsuario())
                + " where id_usuario = "+getId_usuario()+"";
    int n = JOptionPane.showConfirmDialog(
        null,
        "ESTÁ SEGURO DE ACTUALIZAR USUARIO?",
        "PREGUNTA",
        JOptionPane.YES_NO_OPTION);
    if(n==JOptionPane.YES_OPTION){
        Sentencias.executeUpdate(tabla);
        JOptionPane.showMessageDialog(null, "SE ACTUALIZÓ CORRECTAMENTE","ACTUALIZAR", JOptionPane.INFORMATION_MESSAGE);
        Sentencias.close();
    }
    else {
        Sentencias.close();
        JOptionPane.showMessageDialog(null, "NO SE ACTUALIZARON LOS REGISTROS","CONFIRMACIÓN", JOptionPane.INFORMATION_MESSAGE);
    }
}
```

```java
//Cuando es llamado el método con el tipo de perfil y el codigo de usuario, elimina el usuario
public void eliminarUsuario(String tipo,int id)throws SQLException{
    Statement Sentencias = null;
    Sentencias = cone.getConnection().createStatement();
    String tabla = "delete from usuario where id_usuario="+id+" and alias='"+trim(tipo)+"'";
    int n = JOptionPane.showConfirmDialog(
        null,
        "ESTÁ SEGURO DESEA ELIMINAR?",
        "PREGUNTA",
        JOptionPane.YES_NO_OPTION);

    if(n==JOptionPane.YES_OPTION){
        Sentencias.executeUpdate(tabla);
        JOptionPane.showMessageDialog(null, "SE ELIMINÓ CORRECTAMENTE","ELIMINAR", JOptionPane.INFORMATION_MESSAGE);
        Sentencias.close();
    }
    else {
        Sentencias.close();
        JOptionPane.showMessageDialog(null, "NO SE ELIMINA REGISTROS","CONFIRMACIÓN", JOptionPane.INFORMATION_MESSAGE);
    }

}
```

3. **Proveedores:** Clase encargada de crear, modificar y eliminar proveedores

```java
public class Proveedor {
    private String nombreProveedor,rucProveedor,direccionProveedor,telefonoProveedor,emailProveedor,detalleProveedor;
    private int id_proveedor,rucProveedor2;

    conexion cone = new  conexion();
    //CONSTRUCTOR VACIO
    public Proveedor() {
    }

    //CONTRUCTOR CON TODAS LAS VARIABLES
    public Proveedor(String nombreProveedor, String rucProveedor, String direccionProveedor, String telefonoProveedor, String emailProveedor, String detall
        this.nombreProveedor = nombreProveedor;
        this.rucProveedor = rucProveedor;
        this.direccionProveedor = direccionProveedor;
        this.telefonoProveedor = telefonoProveedor;
        this.emailProveedor = emailProveedor;
        this.detalleProveedor = detalleProveedor;
        this.id_proveedor = id_proveedor;
    }


    public String getNombreProveedor() {
        return nombreProveedor;
    }

    public void setNombreProveedor(String nombreProveedor) {
        this.nombreProveedor = nombreProveedor;
    }

    public String getRucProveedor() {
        return rucProveedor;
    }

    public void setRucProveedor(String rucProveedor) {
        this.rucProveedor = rucProveedor;
    }

    public String getDireccionProveedor() {
        return direccionProveedor;
    }

    public void setDireccionProveedor(String direccionProveedor) {
        this.direccionProveedor = direccionProveedor;
    }

    public String getTelefonoProveedor() {
        return telefonoProveedor;
    }

    public void setTelefonoProveedor(String telefonoProveedor) {
        this.telefonoProveedor = telefonoProveedor;
    }

    public String getEmailProveedor() {
        return emailProveedor;
    }

    public void setEmailProveedor(String emailProveedor) {
        this.emailProveedor = emailProveedor;
    }

    public String getDetalleProveedor() {
        return detalleProveedor;
    }

    public void setDetalleProveedor(String detalleProveedor) {
        this.detalleProveedor = detalleProveedor;
    }

    public int getId_proveedor() {
        return id_proveedor;
    }

    public void setId_proveedor(int id_proveedor) {
        this.id_proveedor = id_proveedor;
    }

    //Consulta el proveedor por el id de proveedor
    public void consultaProveedor(int codigo)throws SQLException{
        Statement Sentencias = null;
        Sentencias = cone.getConnection().createStatement();
        ResultSet tabla = Sentencias.executeQuery("SELECT id_proveedor, nombreproveedor, rucproveedor, direccion, telefono, email, detalle FROM proveedor wh
        while(tabla.next()){
            id_proveedor=Integer.parseInt(tabla.getString(1));
            nombreProveedor=tabla.getString(2);
            rucProveedor=tabla.getString(3);
            direccionProveedor=tabla.getString(4);
            telefonoProveedor=tabla.getString(5);
            emailProveedor=tabla.getString(6);
            detalleProveedor=tabla.getString(7);
        }
        Sentencias.close();
    }
```

```
//Método que actualiza los datos de proveedor
public void actualizarProveedor()throws SQLException{
    Statement Sentencias = null;
    Sentencias = cone.getConnection().createStatement();
    String tabla = "UPDATE proveedor "
            + "SET nombreproveedor='"+trim(getNombreProveedor().toUpperCase())+"', rucproveedor='"+trim(getRucProveedor())+"', direccion='"+trim(getDir
            + "WHERE id_proveedor="+getId_proveedor()+"";
    int n = JOptionPane.showConfirmDialog(
        null,
        "ESTÁ SEGURO DE ACTUALIZAR PROVEEDOR?",
        "PREGUNTA",
        JOptionPane.YES_NO_OPTION);

    if(n==JOptionPane.YES_OPTION){
        Sentencias.executeUpdate(tabla);
        JOptionPane.showMessageDialog(null, "SE ACTUALIZÓ CORRECTAMENTE PROVEEDOR","ACTUALIZAR", JOptionPane.INFORMATION_MESSAGE);
        Sentencias.close();
    }
    else {
        Sentencias.close();
        JOptionPane.showMessageDialog(null, "NO SE ACTUALIZARON LOS REGISTROS","CONFIRMACIÓN", JOptionPane.INFORMATION_MESSAGE);
    }
}
```

```
//Método que elimina proveedor por el id.
 public void eliminarProveedor(int id)throws SQLException{
   Statement Sentencias = null;
   Sentencias = cone.getConnection().createStatement();
   String tabla = "delete from proveedor where id_proveedor="+id+"";
   int n = JOptionPane.showConfirmDialog(
       null,
       "ESTÁ SEGURO DESEA ELIMINAR PROVEEDOR?",
       "PREGUNTA",
       JOptionPane.YES_NO_OPTION);

   if(n==JOptionPane.YES_OPTION){
       Sentencias.executeUpdate(tabla);
       JOptionPane.showMessageDialog(null, "SE ELIMINÓ CORRECTAMENTE","ELIMINAR", JOptionPane.INFORMATION_MESSAGE);
       Sentencias.close();
   }
   else {
       Sentencias.close();
       JOptionPane.showMessageDialog(null, "NO SE ELIMINA REGISTROS","CONFIRMACIÓN", JOptionPane.INFORMATION_MESSAGE);
   }


}
```

```
public void crearProveedor()throws SQLException{
  Statement Sentencias = null;
  Sentencias = cone.getConnection().createStatement();
  String tabla = "INSERT INTO proveedor(" +
  "nombreproveedor, rucproveedor, direccion, telefono, email, detalle)" +
  "VALUES ('"+trim(getNombreProveedor())+"','"+trim(getRucProveedor())+"','"+trim(getDireccionProveedor())+"','"+trim(getTelefonoProveedor())+"','"+trim(
      int n = JOptionPane.showConfirmDialog(
      null,
      "ESTÁS SEGURO QUE DESEA CREAR UN PROVEEDOR?",
      "PREGUNTA",
      JOptionPane.YES_NO_OPTION);
  if(n==JOptionPane.YES_OPTION){
      Sentencias.executeUpdate(tabla);
      JOptionPane.showMessageDialog(null, "SE CREÓ CORRECTAMENTE NUEVO PROVEEDOR","CREAR", JOptionPane.INFORMATION_MESSAGE);
      Sentencias.close();
  }
  else {
      JOptionPane.showMessageDialog(null, "NO SE CREA PROVEEDOR","CONFIRMACIÓN", JOptionPane.INFORMATION_MESSAGE);
  }
}
```

```
//Consulta en la base de datos si existe otro numero de cedula retornando un boolean
 public boolean consultaCedula(String cedula)throws SQLException{
       Statement Sentencias = null;
       Sentencias = cone.getConnection().createStatement();
       ResultSet tabla = Sentencias.executeQuery("SELECT count(rucproveedor) FROM proveedor where rucproveedor='"+cedula+"';");
       while(tabla.next()){
           rucProveedor2=Integer.parseInt(tabla.getString(1));
       }
       Sentencias.close();
       if(rucProveedor2==0) return false;
       else return true;
   }

}
```

4. **Repuestos:** Clase encargada de crear, modificar y eliminar repuestos.

```java
/**
 *
 * @author Edison Molina
 */
public class Repuesto {
    private String codigoRepuesto,aplicacionRepuesto,descripcionRepuesto,detalleRepuesto;
    private double preciounitarioRepuesto,precioVentaRepuesto;

    conexion cone = new  conexion();

    public Repuesto() {
    }

    public Repuesto(String codigoRepuesto, String aplicacionRepuesto, String descripcionRepuesto, String detalleRepuesto, double preciounitarioRepuesto,
        this.codigoRepuesto = codigoRepuesto;
        this.aplicacionRepuesto = aplicacionRepuesto;
        this.descripcionRepuesto = descripcionRepuesto;
        this.detalleRepuesto = detalleRepuesto;
        this.preciounitarioRepuesto = preciounitarioRepuesto;
        this.precioVentaRepuesto = precioVentaRepuesto;
    }


    public String getCodigoRepuesto() {
        return codigoRepuesto;
    }

    public void setCodigoRepuesto(String codigoRepuesto) {
        this.codigoRepuesto = codigoRepuesto;
    }

    public String getAplicacionRepuesto() {
        return aplicacionRepuesto;
    }

    public void setAplicacionRepuesto(String aplicacionRepuesto) {
        this.aplicacionRepuesto = aplicacionRepuesto;
    }

    public String getDescripcionRepuesto() {
        return descripcionRepuesto;
    }

    public void setDescripcionRepuesto(String descripcionRepuesto) {
        this.descripcionRepuesto = descripcionRepuesto;
    }

    public String getDetalleRepuesto() {
        return detalleRepuesto;
    }

    public void setDetalleRepuesto(String detalleRepuesto) {
        this.detalleRepuesto = detalleRepuesto;
    }

    public double getPreciounitarioRepuesto() {
        return Math.rint(preciounitarioRepuesto*100)/100;
    }

    public double getPrecioVentaRepuesto() {
        return Math.rint(precioVentaRepuesto*100)/100;
    }

    public void setPrecioVentaRepuesto(double precioVentaRepuesto) {
        this.precioVentaRepuesto = precioVentaRepuesto;
    }

    //cosultaRepuesto trae los datos de la base de datos producto según el codigo y los guarda en las variables.
    public DefaultComboBoxModel mostrarAplicacion() throws SQLException{
        DefaultComboBoxModel lm = new DefaultComboBoxModel();
        Statement Sentencias = null;
        Sentencias = cone.getConnection().createStatement();
        ResultSet tabla = Sentencias.executeQuery("select aplicacion from producto group by aplicacion order by aplicacion");
        while(tabla.next()){
            lm.addElement(tabla.getString(1));
        }
        Sentencias.close();
        return lm;
    }
    //actualizaRepuesto al ser llamado el método actualiza los repuestos
    public void actualizarRepuesto()throws SQLException{
        Statement Sentencias = null;
        Sentencias = cone.getConnection().createStatement();
        String tabla = "update producto set aplicacion='"+trim(getAplicacionRepuesto())+"',descripcion='"+getDescripcionRepuesto()+"',detalle='"+getDetalle
                + " where codigo = '"+trim(getCodigoRepuesto())+"'";

        int n = JOptionPane.showConfirmDialog(
            null,
            "ESTÁ SEGURO DE ACTUALIZAR REPUESTO?",
            "PREGUNTA",
            JOptionPane.YES_NO_OPTION);

        if(n==JOptionPane.YES_OPTION){
            Sentencias.executeUpdate(tabla);
            JOptionPane.showMessageDialog(null, "SE ACTUALIZÓ CORRECTAMENTE REPUESTO","ACTUALIZAR", JOptionPane.INFORMATION_MESSAGE);
            Sentencias.close();
```

```
//eliminarrepuesto es la encarga mediante el codigo eliminar el producto
public void eliminarRepuesto(String Codigo)throws SQLException{
     Statement Sentencias = null;
    Sentencias = cone.getConnection().createStatement();
    String tabla = "delete from producto where codigo='"+Codigo+"'";
    String tabla1 = "delete from kardex where codigo='"+Codigo+"'";
    int n = JOptionPane.showConfirmDialog(
        null,
        "ESTÁ SEGURO DESEA ELIMINAR REPUESTO?",
        "PREGUNTA",
        JOptionPane.YES_NO_OPTION);

    if(n==JOptionPane.YES_OPTION){
        Sentencias.executeUpdate(tabla);
        Sentencias.executeUpdate(tabla1);
        JOptionPane.showMessageDialog(null, "SE ELIMINÓ CORRECTAMENTE RESPUESTO","ELIMINAR", JOptionPane.INFORMATION_MESSAGE);
        Sentencias.close();
    }
    else {
        Sentencias.close();
        JOptionPane.showMessageDialog(null, "NO SE ELIMINA REGISTROS","CONFIRMACIÓN", JOptionPane.INFORMATION_MESSAGE);
    }
}
```

```
//crearRepuesto es la encargada de crear el repuesto validando que ya no exista el código
public void crearRepuesto()throws SQLException{
    Statement Sentencias = null;
    Sentencias = cone.getConnection().createStatement();
    if (validarCodigo(armarCodigo(getAplicacionRepuesto(),getDescripcionRepuesto()))==0){   //revisa duplicados en la base de datos
        String tabla= "insert into producto values('"+armarCodigo(getAplicacionRepuesto(),getDescripcionRepuesto())+"','"+getAplicacionRepuesto()+"','"+
         +getDetalleRepuesto()+"','"+getPreciounitarioRepuesto()+","+getPrecioVentaRepuesto()+")";
        String tabla2="insert into kardex (codigo,stockminimo,stockmaximo,stockactual)values('"+armarCodigo(getAplicacionRepuesto(),getDescripcionRepue

        int n = JOptionPane.showConfirmDialog(
            null,
            "ESTÁS SEGURO QUE DESEA CREAR UN REPUESTO NUEVO?",
            "PREGUNTA",
            JOptionPane.YES_NO_OPTION);
        if(n==JOptionPane.YES_OPTION){
            Sentencias.executeUpdate(tabla);
            Sentencias.executeUpdate(tabla2);
            JOptionPane.showMessageDialog(null, "SE CREÓ CORRECTAMENTE NUEVO REPUESTO","CREAR", JOptionPane.INFORMATION_MESSAGE);
            Sentencias.close();
        }
        else {
            JOptionPane.showMessageDialog(null, "NO SE CREA NUEVO REPUESTO","CONFIRMACIÓN", JOptionPane.INFORMATION_MESSAGE);
        }
    }
        else
         JOptionPane.showMessageDialog(null, "CODIGO DE REPUESTO YA EXISTE","CONFIRMACIÓN", JOptionPane.INFORMATION_MESSAGE);
}
```

```
//esté metodo es el encargado de armar el código con las 3 primeras letras de aplicación y detalle
public String armarCodigo(String uno,String dos) throws SQLException{
    int numeroMayor=0;
    String nombreCodigo,numero;

    String union = (uno.substring(0,3)+dos.substring(0,3)).toUpperCase();//Uné los tres primeros caracteres de cada string para armar el código
    Statement Sentencias = null;
    Sentencias = cone.getConnection().createStatement();
    ResultSet tabla = Sentencias.executeQuery("select max(SUBSTRING(codigo, 7,3)) from producto where codigo like '"+union+"%'");
     while(tabla.next()){
        if(tabla.getString(1)!=null) numeroMayor=Integer.parseInt(tabla.getString(1));
        else numeroMayor=0;
    }
     Sentencias.close();
    numeroMayor=numeroMayor+1;
    numero=String.format("%03d", numeroMayor); //numero de 3 digitos
    nombreCodigo=union+numero;//une los string anteriores con el numero para generar un código unico
    return nombreCodigo;
}
}
```

7.2 Base de datos

En la base de datos encontramos las siguientes estructuras:

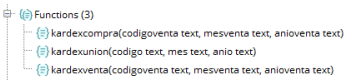
1. Tablas

2. Vistas


```
CREATE OR REPLACE VIEW public.vista_reporte_compra AS
 SELECT v.id_compra,
    pv.nombreproveedor,
    pv.rucproveedor,
    h.fechaemision,
    v.codigo,
    v.aplicacion,
    v.descripcion,
    v.detalle,
    v.costo,
    v.precioventa,
    v.cantidad,
    v.importe,
    h.subtotal,
    (h.subtotal * p.iva::double precision / 100::double precision)::numeric(38,2) AS "IVA",
    h.total
   FROM compra v,
    historialcompra h,
    proveedor pv,
    parametros p
  WHERE h.id_compra = v.id_compra AND h.id_proveedor = pv.id_proveedor;

ALTER TABLE public.vista_reporte_compra
    OWNER TO postgres;
```

3. Funciones




```
CREATE OR REPLACE FUNCTION public.kardexcompra(
    codigoventa text,
    mesventa text,
    anioventa text)
    RETURNS void
    LANGUAGE 'plpgsql'

    COST 100
    VOLATILE
AS $BODY$

begin
    insert into reportekardex select sum(cantidad),max(costo),fechaemision,'1',
    $1,$2,$3
    from vista_reporte_compra
    where TO_CHAR(fechaemision,'MM')=$2 and TO_CHAR(fechaemision,'YYYY')=$3 and codigo=$1
    group by fechaemision
    order by fechaemision;
end

$BODY$;

ALTER FUNCTION public.kardexcompra(text, text, text)
    OWNER TO postgres;
```

8 Glosario

- **JDK** Java Development Kit, software que provee herramientas para desarrollar programas en Java.
- **PgAdmin** Herramienta para administrar bases de datos PostgreSQL.
- **SafePoint** Programa diseñado para el manejo de Bodega para la empresa CarPoint.

Anexo 5. Diccionario de Datos

NombreTabla	Esquema	Tipo
Cliente	public	BASE TABLE
Compra	public	BASE TABLE
Devolución	public	BASE TABLE
Historialcompra	public	BASE TABLE
Kárdex	public	BASE TABLE
Ordentrabajo	public	BASE TABLE
Parámetros	public	BASE TABLE
Perfil	public	BASE TABLE
Producto	public	BASE TABLE
Proveedor	public	BASE TABLE
Reportekardex	public	BASE TABLE
Usuario	public	BASE TABLE
Venta	public	BASE TABLE
vista_kardex	public	VIEW
vista_reporte_compra	public	VIEW
vista_reporte_venta	public	VIEW

TipoRestricción	ColumnaNombre	RestriccionNombre	RestriccionReferencia	TablaReferencia	ColumnaReferencia
PRIMARY KEY	id_compra	historial_pkey			
PRIMARY KEY	id_kardex	kardex_pkey			
PRIMARY KEY	id_perfil	perfil_pkey			
PRIMARY KEY	Código	producto_pkey			
PRIMARY KEY	id_proveedor	proveedor_pkey			
PRIMARY KEY	id_usuario	usuario_pkey			

PRIMARY KEY	id_orden	orden_pkey			
PRIMARY KEY	id_cliente	cliente_pkey			
FOREIGN KEY	Código	fk_id_codigo	producto_pkey	producto	codigo
FOREIGN KEY	id_perfil	fk_id_perfil	perfil_pkey	perfil	id_perfil

Catalogo	NombreFuncion	Esquema	Tipo
Proyecto	kardexunion	public	FUNCTION
Proyecto	kardexventa	public	FUNCTION
Proyecto	kardexcompra	public	FUNCTION