****

UNIVERSIDAD TÉCNICA DE ORURO

FACULTAD NACIONAL DE INGENIERÍA

INGENIERÍA DE INFORMATICA

**SISTEMA DE INFORMACION DE VENTA**



**CASO DE ESTUDIO:** FERRETERIA “FLOWERS”

**ESTUDIANTE:**

UNIV. IVAN FLORES FLORES

**DOCENTE:**

ING. JUAN CARLOS VALLEJOS PANIAGUA

ORURO-BOLIVIA

**CAPITULO I**

1. INTRODUCCIÓN

Actualmente la importancia del manejo de información se ha convertido en una necesidad en el ámbito comercial y empresarial, por tal motivo tener un sistema automatizado, para manejarla de forma adecuada es muy importante.

La implantación de un software de acuerdo a las necesidades generales de los usuarios, del sistema y minimizar los tiempos empleados en el proceso manual, facilita el trabajo para todo el personal.

Por esta razón, una aplicación diseñada hacia el usuario deberá poseer aspectos de uso y seguridad principalmente para asegurar los datos pertenecientes a la empresa.

En los procesos de venta de artículos dirigidos hacia la ciudadanía, es de mucha importancia tener un sistema para ayudar en los registros de datos, realizar el inventario de productos existentes en un almacén, realizar la venta o imprimir una factura, estos ocasionan retrasos que podrían influir de forma insignificante o de manera considerable en la economía de la empresa.

Por lo mencionado anteriormente es de mucho beneficio para LA FERRETERIA “FLORES” el diseño de un sistema y la implementación dirigido al proceso de venta de productos de la empresa.

1. ANTECEDENTES

LA FERRETERIA “FLORES” es una microempresa recientemente inaugurada, realiza la venta de productos variados, proporciona a la población una amplia gama de productos de marcas nacionales e importadas, el proceso de ventas y el registro de las mismas son manuales contando con cuadernos para registrar los productos vendidos. El almacén contiene productos que deben verificarse antes de ofrecerse en stock para la venta. Cuenta con estanterías y vitrinas para mantener todo organizado. Y también ofrece con un punto de venta donde se realiza la comercialización de productos y la atención al cliente.

El proceso de venta obedece como ya se mencionó, a un proceso manual donde se listan los productos en una factura o boleta de compra, la cantidad y el precio de venta del producto, se calcula un subtotal y para luego sumar el precio total y cobrarla al cliente.

La ferretería cuenta con un gerente propietario, un cajero encargado de la venta de productos y la emisión de factura al cliente, un encargado de reponer los productos que hacen falta en las estanterías y vitrinas y un encargado de limpieza.

Ahora se pretende desarrollar un sistema de información para propósitos generalmente para ayudar al proceso de venta en LA FERRETERIA “FLORES”.

1. ORGANIGRAMA DE LA INSTITUCIÓN

GERENTE

CLIENTE

ADMINISTRADOR

CAJERO

REALIZAR VENTA

REGISTRO

USUARIO

REALIZAR COMPRA

GESTIONAR PRODUCTO

1. PLANTEAMIENTO DEL PROBLEMA

¿COMO MINIMIZAR EL TIEMPO DE ATENCIÓN AL CLIENTE Y QUE ESTE, PUEDA REALIZAR SU COMPRA MEDIANTE EL SISTEMA DE VENTAS PARA OBTENER UN SERVICIO RÁPIDO Y CONFIABLE EN LA FERRETERIA “FLORES”?

1. OBJETO DE ESTUDIO

El desarrollo de un sistema de Ventas en línea o en atención en el cajero mediante una aplicación web, reduciendo el tiempo de atención en la venta de productos y asegurando la información de LA FERRETERIA “FLORES”.

1. OBJETIVOS
   1. OBJETIVO GENERAL

Desarrollar un sistema de ventas LA FERRETERIA “FLORES” que permita la venta de productos en línea y por cajeros controlando y actualizando los registros de los productos, de forma segura.

* 1. OBJETIVOS ESPECÍFICOS
* Realizar un análisis de requerimientos para conocer las necesidades de los usuarios del sistema.
* Realizar el módulo de venta de productos.
* Realizar el módulo de gestión de productos.
* Realizar los reportes de los productos.
* Realizar interfaces amigables para los usuarios del sistema.
* Realizar interfaces amigables para que los clientes puedan comprar en línea.
* Realizar la base de datos para el manejo seguro de la información.
* Realizar las pruebas para evitar errores en el funcionamiento del sistema.

1. CAMPO DE ACCIÓN

Se realiza el proceso de ventas por medio de un cuaderno en LA FERRETERIA “FLORES” donde se desarrolló un sistema de información y la implantación del mismo luego del desarrollo del software. Para que la información sea más confiable y segura tanto para el vendedor como el comprador.

1. JUSTIFICACIONES
   1. TÉCNICA

El desarrollo del software seguirá la línea del proceso de desarrollo UML-RUP, la codificación se desarrollara en el framework Codeigniter para el desarrollo de aplicaciones en php que utiliza el MVC. Los cuáles serán de gran aporte a la práctica de proyectos de desarrollo de software.

* 1. SOCIAL

Los beneficiarios del presente proyecto serán LA FERRETERIA”FLORES” y los clientes, quienes acuden a realizar sus necesidad comerciales al adquirir un producto que está en la casa comercial.

* 1. ALCANCES

Se obtendrá un producto software que satisfaga las necesidades básicas en el proceso de ventas en LA FERRETERIA “FLORES”, desarrollando un sistema de ventas con módulos funcionales, facilitando de este modo el trabajo más efectivo por parte del vendedor como del comprador.

1. INGENIERÍA DEL PROYECTO

* Se realizará una visita al negocio y una entrevista con encargados de los procesos desarrollados en la ferretería, para obtener las necesidades de los clientes y los encargados en ventas. En el manejo y aplicabilidad del instrumento o herramienta informática empleada en la casa comercial, ofreciendo de esta forma una atención tecnológica, confiable.
* Viendo el proceso detallado de la venta de productos de LA FERRETERIA “FLORES”.
* Apoyado por el framework Codeigniter se logrará realizar los módulos funcionales, para los usuarios.
* Realizar la base de datos para el manejo de información, tanto de los productos en oferta, como la accesibilidad de los usuarios que acuden a la casa comercial en este casa la ferretería.
* Apoyándonos con phpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando Internet. Logrando crear una base de datos para el manejo de la información.
* Con la ayuda de métodos de pruebas (Pruebas de caja negra, Pruebas alfa, Pruebas beta, etc.)

**CAPITULO II**

**MARCO TEÓRICO**

1. MODELO, VISTA, CONTROLADOR (MVC)

El MVC o Modelo-Vista-Controlador es un patrón de arquitectura de software que, utilizando 3 componentes (Vistas, Models y Controladores) separa la lógica de la aplicación de la lógica de la vista en una aplicación. Es una arquitectura importante puesto que se utiliza tanto en componentes gráficos básicos hasta sistemas empresariales; la mayoría de los frameworks modernos utilizan MVC (o alguna adaptación del MVC) para la arquitectura, entre ellos podemos mencionar a Ruby on Rails, Django, AngularJS y muchos otros más.

En este método de análisis y diseño se crea un conjunto de modelos utilizando una notación acordada como, por ejemplo, el lenguaje unificado de modelado (UML).

La mayoría de las aplicaciones desarrolladas hoy en día son las aplicaciones llamadas Web, es decir, aquellas que tienen como elemento significativo de su arquitectura un navegador y un protocolo de comunicación HTTP. Cuando capacitamos a la gente en arquitectura y patrones, buscamos que el alumno comprenda las formas de elaborar este tipo de aplicaciones, como su ubicación en algunos de los patrones de arquitectura Web: “Cliente Delgado Web”, “Cliente Robusto Web” o “Reparto Web”.

En 1998, Jim Conallen definió una extensión a la que denominó WAE (Web Application Extension) para UML. Esta extensión es la convención más difundida y aceptada hasta nuestros días y podríamos decir que define el estándar de facto. En esta entrega, formada por dos artículos, presentaremos los elementos que definen el 20-80 en el modelado de aplicaciones Web usando la WAE. El foco de este artículo es la WAE en los diagrama de clases.

Los puntos anteriores son sólo para proveer background, y utilizar las referencias ahora que vamos a explicar qué es.

Antes que nada, me gustaría mencionar por qué se utiliza el MVC, la razón es que nos permite separar los componentes de nuestra aplicación dependiendo de la responsabilidad que tienen, esto significa que cuando hacemos un cambio en alguna parte de nuestro código, esto no afecte otra parte del mismo. Por ejemplo, si modificamos nuestra Base de Datos, sólo deberíamos modificar el modelo que es quién se encarga de los datos y el resto de la aplicación debería permanecer intacta. Esto respeta el principio de la responsabilidad única. Es decir, una parte de tu código no debe de saber qué es lo que hace toda la aplicación, sólo debe de tener una responsabilidad.

En web, el MVC funcionaría así. Cuando el usuario manda una petición al navegador, digamos quiere ver el curso de AngularJS, el controlador responde a la solicitud, porque él es el que controla la lógica de la app, una vez que el controlador nota que el usuario solicitó el curso de Angular, le pide al modelo la información del curso.

El modelo, que se encarga de los datos de la app, consulta la base de datos y digamos, obtiene todos los vídeos del curso de AngularJS, la información del curso y el título, el modelo responde al controlador con los datos que pidió (nota como en la imagen las flechas van en ambos sentidos, porque el controlador pide datos, y el modelo responde con los datos solicitados).

Una vez el controlador tiene los datos del curso de AngularJS, se los manda a la vista, la vista aplica los estilos, organiza la información y construye la página que se ve en el navegador.

Resumamos entonces los conceptos.

**Modelo**

Se encarga de los datos, generalmente (pero no obligatoriamente) consultando la base de datos. Actualizaciones, consultas, búsquedas, etc. todo eso va aquí, en el modelo.

**Controlador**

Se encarga de... controlar, recibe las órdenes del usuario y se encarga de solicitar los datos al modelo y de comunicárselos a la vista.

**Vistas**

Son la representación visual de los datos, todo lo que tenga que ver con la interfaz gráfica va aquí. Ni el modelo ni el controlador se preocupan de cómo se verán los datos, esa responsabilidad es únicamente de la vista.

1. LENGUAJE UNIFICADO DE MODELADO (UML)

El **lenguaje unificado de modelado** (**UML**, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de [software](https://es.wikipedia.org/wiki/Software) más conocido y utilizado en la actualidad; está respaldado por el [Object Management Group](https://es.wikipedia.org/wiki/Object_Management_Group) (OMG).

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

Es importante remarcar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Se puede aplicar en el desarrollo de software gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el [Proceso Unificado Racional](https://es.wikipedia.org/wiki/Proceso_Unificado_Racional), Rational Unified Process o [RUP](https://es.wikipedia.org/wiki/RUP)), pero no especifica en sí mismo qué metodología o proceso usar.

UML no puede compararse con la [programación estructurada](https://es.wikipedia.org/wiki/Programaci%C3%B3n_estructurada), pues UML significa Lenguaje Unificado de Modelado, no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que programación estructurada es una forma de programar como lo es la orientación a objetos, la programación orientada a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML solo para lenguajes orientados a objetos.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

En el año [2012](https://es.wikipedia.org/wiki/2012) se actualizó la norma a la última versión definitiva disponible en ese momento, la 2.4.1, dando lugar a las normas ISO/IEC 19505-1

**Historia**

**Antes de UML 1.x**

Después de que la Rational Software Corporation contratara a James Rumbaugh de [General Electric](https://es.wikipedia.org/wiki/General_Electric), en [1994](https://es.wikipedia.org/wiki/1994), la compañía se convirtió en la fuente de los dos esquemas de modelado orientado a objetos más populares de la época: Object-Modeling Technique (OMT) de Rumbaugh, que era mejor para [análisis orientado a objetos](https://es.wikipedia.org/wiki/An%C3%A1lisis_y_dise%C3%B1o_orientado_a_objetos), y el Método Booch (de Grady Booch) que era mejor para el [diseño orientado a objetos](https://es.wikipedia.org/wiki/Dise%C3%B1o_orientado_a_objetos). Poco después se les unió [Ivar Jacobson](https://es.wikipedia.org/wiki/Ivar_Jacobson), el creador del método de ingeniería de software orientado a objetos. Jacobson se unió a Rational, en [1995](https://es.wikipedia.org/wiki/1995), después de que su compañía Objectory AB fuera comprada por Rational. Los tres metodologistas eran conocidos como los Tres Amigos, porque se sabía de sus constantes discusiones sobre las prácticas metodológicas.

En [1996](https://es.wikipedia.org/wiki/1996) Rational concluyó que la abundancia de lenguajes de modelado estaba alentando la adopción de la tecnología de objetos, y para orientarse hacia un método unificado, encargaron a los Tres Amigos que desarrollaran un "lenguaje unificado de modelado" abierto. Se consultó con representantes de compañías competidoras en el área de la tecnología de objetos durante la OOPSLA '96; eligieron "cajas" para representar clases en lugar de la notación de Booch que utilizaba símbolos de "nubes". Bajo la dirección técnica de los Tres Amigos (Rumbaugh, Jacobson y Booch) fue organizado un consorcio internacional llamado UML Partners en [1996](https://es.wikipedia.org/wiki/1996) para completar las especificaciones del UML, y para proponerlo como una respuesta al OMG RFP. El borrador de la especificación UML 1.0 de UML Partners fue propuesto a la OMG en [enero de 1997](https://es.wikipedia.org/wiki/Enero_de_1997). Durante el mismo mes, la UML Partners formó una Fuerza de Tarea Semántica, encabezada por Cris Kobryn y administrada por Ed Eykholt, para finalizar las semánticas de la especificación y para integrarla con otros esfuerzos de estandarización. El resultado de este trabajo, el UML 1.1, fue presentado ante la OMG en [agosto de 1997](https://es.wikipedia.org/wiki/Agosto_de_1997) y adoptado por la OMG en [noviembre de 1997](https://es.wikipedia.org/wiki/Noviembre_de_1997).

**UML 1.x**

Como notación de modelado, la influencia de la OMT domina UML (por ejemplo, el uso de rectángulos para clases y objetos). Aunque se quitó la notación de "nubes" de Booch, sí se adoptó la capacidad de Booch para especificar detalles de diseño en los niveles inferiores. La notación de "Casos de Uso" del *Objectory* y la notación de componentes de Booch fueron integradas al resto de la notación, pero la integración semántica era relativamente débil en UML 1.1, y no se arregló realmente hasta la revisión mayor de UML 2.0.

Conceptos de muchos otros métodos orientados a objetos (MOO) fueron integrados superficialmente en UML con el propósito de hacerlo compatible con todos los MOO. Además, el grupo tomó en cuenta muchos otros métodos de la época, con el objetivo de asegurar amplia cobertura en el dominio de los sistemas en tiempo real. Como resultado, UML es útil en una gran variedad de problemas de ingeniería, desde procesos sencillos y aplicaciones de solamente un usuario a sistemas concurrentes y distribuidos.

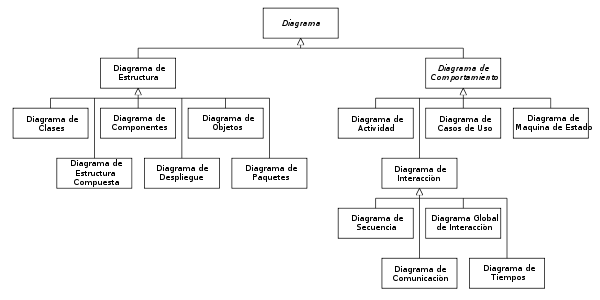
**UML 2.x**

UML ha madurado considerablemente desde UML 1.1, varias revisiones menores (UML 1.3, 1.4 y 1.5) han corregido defectos y errores de la primera versión de UML. A estas le ha seguido la revisión mayor UML 2.0 que fue adoptada por el OMG en [2005](https://es.wikipedia.org/wiki/2005).

Aunque UML 2.1 nunca fue lanzado como una especificación formal, las versiones 2.1.1 y 2.1.2, aparecieron en [2007](https://es.wikipedia.org/wiki/2007), seguidas por UML 2.2 en [febrero de 2009](https://es.wikipedia.org/wiki/Febrero_de_2009). UML 2.3 fue lanzado oficialmente en [mayo de 2010](https://es.wikipedia.org/wiki/Mayo_de_2010). UML 2.4.1 fue lanzado oficialmente en [agosto de 2011](https://es.wikipedia.org/wiki/Agosto_de_2011). UML 2.5 fue lanzado en [octubre de 2012](https://es.wikipedia.org/wiki/Octubre_de_2012) como una versión "En proceso" que fue formalmente liberada en [junio de 2015](https://es.wikipedia.org/wiki/Junio_de_2015).

**Tipos de diagramas en UML 2.5**

Existen dos clases principales de tipos de diagramas: diagramas *estructurales* y diagramas de *comportamiento*. Estos últimos incluyen varios que representan diferentes aspectos de las interacciones. Estos diagramas pueden ser categorizados jerárquicamente como se muestra en el siguiente diagrama de clases:

[](https://commons.wikimedia.org/wiki/File:Uml_diagram-es.svg)

**Estructurales**

Muestran la estructura estática de los objetos en un sistema.

[**Diagrama de clases**](https://es.wikipedia.org/wiki/Diagrama_de_clases) Los diagramas de clase son, sin duda, el tipo de diagrama UML más utilizado. Es el bloque de construcción principal de cualquier solución orientada a objetos. Muestra las clases en un sistema, atributos y operaciones de cada clase y la relación entre cada clase. En la mayoría de las herramientas de modelado, una clase tiene tres partes, nombre en la parte superior, atributos en el centro y operaciones o métodos en la parte inferior. En sistemas grandes con muchas clases relacionadas, las clases se agrupan para crear diagramas de clases. Las Diferentes relaciones entre las clases se muestran por diferentes tipos de flechas.

[**Diagrama de componentes**](https://es.wikipedia.org/wiki/Diagrama_de_componentes)Un diagrama de componentes muestra la relación estructural de los componentes de un sistema de software. Estos se utilizan principalmente cuando se trabaja con sistemas complejos que tienen muchos componentes. Los componentes se comunican entre sí mediante interfaces. Las interfaces se enlazan mediante conectores.

[**Diagrama de despliegue**](https://es.wikipedia.org/wiki/Diagrama_de_despliegue) Un diagrama de despliegue muestra el hardware de su sistema y el software de ese hardware. Los diagramas de implementación son útiles cuando la solución de software se despliega en varios equipos, cada uno con una configuración única.

[**Diagrama de objetos**](https://es.wikipedia.org/wiki/Diagrama_de_objetos) Los diagramas de objetos, a veces denominados diagramas de instancia, son muy similares a los diagramas de clases. Al igual que los diagramas de clases, también muestran la relación entre los objetos, pero usan ejemplos del mundo real. Se utilizan para mostrar cómo se verá un sistema en un momento dado. Debido a que hay datos disponibles en los objetos, a menudo se utilizan para explicar relaciones complejas entre objetos.

[**Diagrama de paquetes**](https://es.wikipedia.org/wiki/Diagrama_de_paquetes)Como su nombre indica, un diagrama de paquetes muestra las dependencias entre diferentes paquetes de un sistema.

[**Diagrama de perfiles**](https://es.wikipedia.org/w/index.php?title=Diagrama_de_perfiles&action=edit&redlink=1)El diagrama de perfil es un nuevo tipo de diagrama introducido en UML 2. Este es un tipo de diagrama que se utiliza muy raramente en cualquier especificación.

[**Diagrama de estructura compuesta**](https://es.wikipedia.org/wiki/Diagrama_de_estructura_compuesta)Los diagramas de estructura compuesta se utilizan para mostrar la estructura interna de una clase.

**De comportamiento**

Muestran el comportamiento dinámico de los objetos en el sistema.

[**Diagrama de actividades**](https://es.wikipedia.org/wiki/Diagrama_de_actividades) Los diagramas de actividad representan los flujos de trabajo de forma gráfica. Pueden utilizarse para describir el flujo de trabajo empresarial o el flujo de trabajo operativo de cualquier componente de un sistema. A veces, los diagramas de actividad se utilizan como una alternativa a los diagramas de máquina del estado.

[**Diagrama de casos de uso**](https://es.wikipedia.org/wiki/Diagrama_de_casos_de_uso) Como el tipo de diagrama de diagramas UML más conocido, los diagramas de casos de uso ofrecen una visión general de los actores involucrados en un sistema, las diferentes funciones que necesitan esos actores y cómo interactúan estas diferentes funciones. Es un gran punto de partida para cualquier discusión del proyecto, ya que se pueden identificar fácilmente los principales actores involucrados y los principales procesos del sistema.

[**Diagrama de máquina de estados**](https://es.wikipedia.org/w/index.php?title=Diagrama_de_m%C3%A1quina_de_estados&action=edit&redlink=1) Los diagramas de máquina de estado son similares a los diagramas de actividad, aunque las anotaciones y el uso cambian un poco. En algún momento se conocen como diagramas de estados o diagramas de diagramas de estado también. Estos son muy útiles para describir el comportamiento de los objetos que actúan de manera diferente de acuerdo con el estado en que se encuentran en el momento.

**Diagrama de interacción**. Los diagramas de interacción incluyen distintos tipos de diagramas:

[**Diagrama de secuencia**](https://es.wikipedia.org/wiki/Diagrama_de_secuencia) Los diagramas de secuencia en UML muestran cómo los objetos interactúan entre sí y el orden en que se producen esas interacciones. Es importante tener en cuenta que muestran las interacciones para un escenario en particular. Los procesos se representan verticalmente y las interacciones se muestran como flechas. Los diagramas de secuencia de UML forman parte de un modelo UML y solo existen dentro de los proyectos de modelado UML. Para crear un diagrama de secuencia UML, en el menú Arquitectura, haga clic en Nuevo diagrama de capas o UML. Obtenga más información sobre elementos de diagrama de secuencia UML o diagramas de modelado UML en general. Para ver una demostración en vídeo, consulte Esbozar interacciones mediante diagramas de secuencia (2010).

[**Diagrama de comunicación**](https://es.wikipedia.org/wiki/Diagrama_de_comunicaci%C3%B3n) El diagrama de comunicación se llamó diagrama de colaboración en UML 1. Es similar a los diagramas de secuencia, pero el foco está en los mensajes pasados entre objetos.

[**Diagrama de tiempos**](https://es.wikipedia.org/wiki/Diagrama_de_tiempos) Los diagramas de sincronización son muy similares a los diagramas de secuencia. Representan el comportamiento de los objetos en un marco de tiempo dado. Si es solo un objeto, el diagrama es directo, pero si hay más de un objeto involucrado, también se pueden usar para mostrar interacciones de objetos durante ese período de tiempo.

[**Diagrama global de interacciones**](https://es.wikipedia.org/wiki/Diagrama_global_de_interacciones)Los diagramas generales o globales de interacción son muy similares a los diagramas de actividad. Mientras que los diagramas de actividad muestran una secuencia de procesos, los diagramas de interacción muestran una secuencia de diagramas de interacción. En términos simples, pueden llamarse una colección de diagramas de interacción y el orden en que suceden. Como se mencionó anteriormente, hay siete tipos de diagramas de interacción, por lo que cualquiera de ellos puede ser un nodo en un diagrama de vista general de interacción.

**CAPITULO III**

**REQUERIMIENTOS, ANÁLISIS, DISEÑO E IMPLEMENTACIÓN**

1. MODELO CONCEPTUAL

El negocio abre sus puertas al público de 8:30 a.m. hasta las 18:30 p.m. horas de lunes a viernes, además el portal web está activo todo el tiempo. Cuenta con un gerente que es el dueño del negocio está encargado de dar instrucciones a los demás empleados, el Administrador que controla las actividades como horarios de llegada, registro de usuarios, etc.; también se cuenta con una Secretaria que es el actor más importante del negocio ya que es el encargado de controlar la venta de los productos de la ferretería y finalmente la encargada de limpieza siempre debe mantener limpio las oficinas ya que son la presencia de la ferretería.

Elemento fundamental del negocio es el cliente que es lo que hace funcionar el negocio.

1. ESPECIFICACIÓN DE REQUERIMIENTOS
   1. REQUERIMIENTOS FUNCIONALES

* El sistema deberá verificar la autenticidad de los usurarios
* El sistema deberá gestionar los datos de los productos en la base de datos.
* El sistema deberá gestionar la información de nuevos usuarios.
* El sistema deberá gestionar la información de los productos en línea.
* El sistema deberá mostrar los productos y usuarios almacenados en la base de datos.
* El sistema deberá emitir una lista impresa según se requiera.
* El sistema deberá emitir factura de compra de los productos según requiera el cliente.
* El sistema debe contar con conexión a internet para que los clientes realicen sus compras.
* Los clientes deben pagar con tarjeta de crédito.
* Los clientes se van a informar de los diferentes productos y precios que se ofrecen.
* Permite que el usuario no introduzca datos incorrectos en los campos.
  1. REQUERIMIENTOS NO FUNCIONALES
* Seguridad, el sistema contara con nivel de validación de usuarios.
* Facilidad de uso, el sistema será lo más sencillo posible.
* Confiabilidad, grado en el que se espera que un programa cumpla con su función y con la precisión requerida.
* Seguridad.
* Navegadores compatibles.
* Fiabilidad.
* Disponibilidad.
* Usabilidad.

1. REQUERIMIENTOS DE SOFTWARE Y HARDWARE

* Sistema operativo sin discriminación.
* PHP version 5.6
* MySQL (5.1+) a través de los controladores mysql (en desuso), mysqli.
* PC con Procesador Pentium R 2.8 Ghz(mínimo),Memoria: 1 Gb. (mínimo), Dispositivos de salida (Monitor, impresora), Dispositivos de entrada (Teclado, mouse)

1. LISTA DE CLASES

|  |  |
| --- | --- |
| **Clase** | **Atributos** |
| categoria | Idcategoria, descricion |
| Cliente | Idcliente,dni, apellidos, nombres, dirección, teléfono |
| Detalleventa | Iddetalle\_venta, idproducto, preducto\_unitario, cantidad |
| Empleado | Idempleado, estado, dni, apellidos, nombres, user, password,rol |
| Producto | Idproducto, idcategoria, nombres, descripciones, cantidad, precio\_compra, precio\_venta, fecha, imagen |
| Venta | Idventa, idempleado, idcliente, serie, nro\_documento, fecha\_venta |

1. MODELO CONCEPTUAL

idcategoria

descripcion

idproducto

idcategoria

descripciones

Iddetalle\_venta

cantidad

Preducto\_unitario

idventa

idproducto

detalleventa

categoria

producto

cantidad

Precio\_compra

Precio\_venta

nombre

imagen

fecha

venta

idempleado

idventa

idcliente

Fecha\_venta

Nro\_documento

serie

empleado

cliente

estado

direccion

dni

apellidos

nombres

telefono

idcliente

idempleado

apellidos

nombres

dni

user

password

rol

1. MODELO DEL DOMINIO



1. MODELO DEL NEGOCIO



1. CASOS DE USO REALES



|  |  |  |
| --- | --- | --- |
| **Caso de uso** | Validar usuario | |
| **Actores** | Usuario (Empleado, Administrador) | |
| **Resumen** | El caso de uso inicia cuando el usuario desea ingresar al sistema, introduce el usuario y la contraseña asignado, accediendo a la pantalla de carga para posteriormente se habilitan dos pestañas que tienen acceso el usuario. | |
| **Responsabilidad** | Verificar la autenticidad del usuario para acceder al sistema. | |
| **CU asociados** |  | |
| **Precondiciones** |  | |
| **Descripción** | | |
| **Acción Actor(es)**  Pantalla 1 | | **Respuesta del sistema**  Pantalla 2 |
| **C**  **B**  **A** | | |
| **1.** El caso de uso inicia cuando el usuario necesita ingresar al sistema. | |  |
|  | | **2.** Se habilitan las pestañas Gestionar Producto y Gestionar Usuario, en las cuales puedes realizar los cambios en los productos y usuarios. |
| **3.** Ingresa el usuario y la contraseña.  **4.** Realiza clic en el botón ingresar. | |  |
|  | | **5.** El sistema autentica la cuenta con los registros de la base de datos.  **6.** El sistema muestra la pantalla de Inicio al ingresar. |
| **Textos Alternos**  Si la autenticación no es correcta el sistema le vota a inicio sin poder acceder a las pestañas designadas. | | |
| **Requisitos No funcionales** | Seguridad en la aplicación | |
| **Caso de uso** | Gestionar Producto | |
| **Actores** | Usuario (Empleado, Administrador) | |
| **Resumen** | El caso de uso inicia cuando el usuario quiere gestionar el producto debe hacer clip en la pestaña indicada. | |
| **Responsabilidad** | Mostrar las acciones que puede hacer con los productos. | |
| **CU asociados** | Registrar Producto, Modificar Producto, Eliminar Producto | |
| **Precondiciones** | Validar usuario | |
| **Descripción** | | |
| **Acción Actor(es)**  Pantalla 1 | | **Respuesta del sistema**  Pantalla 2 |
| **A**      Pantalla 3  Pantalla 4 | | |
| **1.** El caso de uso inicia cuando el usuario necesita Registrar, editar o eliminar los registros del producto | |  |
|  | | **2.** Ofrece la interfaz (Pantalla 1), en la xaul puedes insertar, modificar, eliminar el producto e imprimir el listado de los productos. |
| **3.** RealizaClic en | |  |
|  | | **4.** El sistema muestra un módulo de modificación de productos. En la (Pantalla 2), presionando en el botón modificar.  **5.** El sistema muestra las opciones de eliminar el producto. |
| **6.** El usuario puede imprimir la lista de los productos | |  |
|  | |  |
| **Textos Alternos** Cada vez que quiera guardar los productos verifica si los datos están correctos, mostrando un mensaje. | | |
| **Requisitos No funcionales** | Usabilidad | |

1. DIAGRAMA DE CLASES DE DISEÑO



1. DIAGRAMA DE SECUENCIA







1. DIAGRAMA DE DESPLIEGUE Y COMUNICACIÓN



**CAPITULO IV**

**PRUEBAS DE CAJA NEGRA**

|  |  |
| --- | --- |
| **Casos de Prueba** | Registrar Usuario |
| **Actores** | Usuario |
| **Propósito** | Verificar los campos ingresados |
| **Entrada**  **C:\Users\CORE-I3\OneDrive\Imágenes\Capturas de pantalla\2019-01-14 (8).png** | Nombre: 1- “ ”  2. ivan  3. ana  4. juan  Contraseña: 1. “ ”  2. 1234  3. flores@4$  4. loque odebo escribiri…) |
| Condición  **Clase validas**   1) carácter >= 3  2) carácter <= 20  7) carácter >= 3  8) carácter <= 20 | **Clases invalidas**  3) > 20 caracteres  4) < 3 caracteres  5) “ ” entrada vacía  6) caracteres especial (+,&,$,/,\*,-, @)  9) > 20 caracteres  10) < 3 caracteres  11) “ ” entrada vacía  12) caracteres especial (+,&,$,/,\*,-, @) |
| **Salida**  Nombre: “ ”  ivan  ana  gato =z>  Contraseña:  “ ”  admin  flores@4$  loque odebo escribiri…) | **Observaciones**  Error 5)  Correcto 1),2)  Correcto 1),2)  Error 4)  Error 11)  Correcto 7),8)  Error 12)  Error 9 |

**CONCLUSIONES**

* Se logró recopilar requerimientos funcionales sobre los procesos de registro de los productos en la FERRETERIA FLORES obteniendo datos para realizar la aplicación web.
* La aplicación web permite el control sobre toda la información de los productos, como de los usuarios del sistema.
* La aplicación tiene una interfaces amigables y de fácil de usar.
* El sistema permite realizar la impresión o guardar de los registrados de venta de los artículos que se encuentran en la base de datos.
* Con la implementación de las pruebas de caja negra en algunos casos de uso se logró minimizar los errores del sistema de registro.