UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

MAESTRÍA EN INFORMÁTICA

TESIS

DESARROLLO DE UN SISTEMA DE VENTAS PARA LA PLANTA DE CRIADERO DE TRUCHAS ARAPA S.A.C. - 2018

PRESENTADA POR:

RENAN ABELARDO PALLI MAMANI

PARA OPTAR EL GRADO ACADEMICO DE:

MAGISTER SCIENTIAE EN INFORMÁTICA MENCIÓN GERENCIA DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES

PUNO, PERÚ

2019

UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

MAESTRÍA EN INFORMÁTICA

TESIS

DESARROLLO DE UN SISTEMA DE VENTAS PARA LA PLANTA DE CRIADERO DE TRUCHAS ARAPA S.A.C. - 2018

PRESENTADA POR:

RENAN ABELARDO PALLI MAMANI

PARA OPTAR EL GRADO ACADÉMICO DE:

MAGISTER SCIENTIAE EN INFORMÁTICA
MENCIÓN EN GERENCIA DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES

APROBADA POR EL SIGUIENTE JURADO:

PRESIDENTE

.....
Dr. LEONEL COYLA IDME

PRIMER MIEMBRO

.....
M.Sc. ERNESTO NAYER TUMI FIGUEROA

SEGUNDO MIEMBRO

.....
M.Sc. LEONID ALEMAN GONZALES

ASESOR DE TESIS

.....
D.Sc. PERCY HUATA PANCA

Puno, 30 de Mayo de 2019

ÁREA: Sistemas inteligentes adaptados con información heterogénea

TEMA: Desarrollo de un sistema de ventas

DEDICATORIA

A Dios por guiar y alumbrar mi vida por el camino del bien y otorgarme el don de vivir y hacer posible la realización de mis más grandes anhelos.

Con mucho cariño e infinita gratitud a mis queridos padres Gabriel e Ida, quienes me ofrecieron su constante apoyo y aliento para seguir adelante en todas las metas que me trace.

A mi hermanita querida, por el respaldo moral que siempre me brinda con inmenso cariño incondicional y por la esperanza que me da en las metas que me trazo.

A mis docentes de la maestría de mi querida Universidad quienes me brindaron su comprensión infinita y sus buenos consejos para encaminar mi vida, a ellos mis más sinceras gratitudes.

AGRADECIMIENTOS

- A la Universidad Nacional del Altiplano, Y escuela de Postgrado, la Maestría en Informática, en la mención de Gerencia de Tecnologías de la Información y Comunicaciones por haberme formado y así poder alcanzar una más de mis metas.

- A mis jurados, asesor y docentes de la escuela de Postgrado: Maestría en Informática, por sus estimadas sugerencias y por haberme transmitido sus conocimientos, y apoyarme en la culminación de mi trabajo de investigación

- Agradezco, a mis padres y a mis amigos de la maestría, y a Director y Asesor de tesis por ayudarme en el área de informática y en mi trabajo de investigación.

ÍNDICE GENERAL

Pág.

CAPÍTULO II

PLANTEAMIENTO DEL PROBLEMA

CAPÍTULO III

MATERIALES Y MÉTODOS

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Pág.

ÍNDICE DE ANEXOS

Pág.

RESUMEN

En esta investigación presentamos el análisis y desarrollo de un sistema de ventas avanzado, empezó a desarrollarse con la importancia que tiene la sistematización y la existencia de un software que permite la disminución de esfuerzo, tiempo, dinero y entre otros factores en beneficio a la empresa y/o usuarios, lo que hace que éstos sean más requeridos. El presente trabajo se realizó en el Distrito de Arapa, Provincia de Azángaro Región Puno, planta de criadero de truchas Arapa S.A.C. el cual no cuenta con un sistema que permita optimizar sus ventas en el mercado, ya que el control de las ventas se efectuaba en cuadernos de apunte o en hojas de cálculo Excel. El objetivo que se alcanzo fue desarrollar un sistema de ventas avanzado que les controle: las ventas, ingresos y egresos que este negocio conlleva, debido a que exportan grandes cantidades de trucha al mercado nacional e internacional. La metodología que se utilizó para el desarrollo del sistema es la metodología de desarrollo XP (Programación extrema), es una metodología muy rápida y ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado. En el cual se ha adaptado los algoritmos de selección en los pesos de las truchas para una mejor toma de decisiones al momento de realizar las ventas; Se utilizó el CakePHP, Framework codeigniter; llegando a concluir que el sistema de ventas fue de gran utilidad para la planta de criadero de truchas Arapa S.A.C.

Palabras clave: avanzado, CakePhP, desarrollo, framework, truchas y ventas

ABSTRACT

In this research we present the analysis and development of an advanced sales system, develop a development and the importance that we have systematization and the existence of software that allows the reduction of effort, time, money and other factors for the benefit of the company and / or users, which makes them more required. The present work was carried out in the District of Arapa, Province of Azángaro Puno Region, Arapa trout breeding plant S.A.C. which does not have a system that allows you to optimize your sales in the market, and the control of sales is done in notebooks or Excel spreadsheets. The objective of which was achieved was to develop an advanced sales system that controls: the sales, revenues and expenses that this business entails, because they export large quantities of trout to the national and international market. The methodology used for the development of the system is based on simplicity, communication and feedback or the reuse of the code. In which the algorithms of selection in the weights of the trout have been adapted for a better decision making at the time of making the sales; We used the CakePHP, Framework codeigniter; Arriving at the conclusion that the sales system was very useful for the trout breeding plant Arapa S.A.C.

Keywords: advanced, CakePhP, development, framework, trout and sales

INTRODUCCIÓN

El desarrollo de sistemas expertos es una herramienta necesaria en las empresas puesto que ayuda a tomar decisiones en beneficio a la empresa para así poder generar más ingresos y/o ganancias, de tal manera que también se mejora el control administrativo, todo esto se hace en tiempo real proporcionando reportes detallados de las ventas, también poder estar inspeccionando el stock de los productos. Los sistemas de ventas expertos pueden ser personalizados para cumplir con las necesidades específicas de una industria por ejemplo si se desea controlar el stock o que el sistema tome decisiones propias en favor de la empresa también hacer reportes, productos, categorías que hay en la empresa y también de los clientes según se requiera; la estructura del trabajo se presenta a continuación:

En el capítulo I se presentan los aspectos generales relacionados con el trabajo de investigación a realizar en este trabajo de tesis.

En el capítulo II se presentan las situaciones del problema que se presentan al momento de realizar las ventas e de aquí donde sale la idea de implementar un sistema de ventas avanzado el cual ayudara en gran magnitud a la empresa.

En el capítulo III mostramos el desarrollo de la metodología Programación Extrema (XP) y empleando el Lenguaje Unificado de Modelamientos (UML).

En el capítulo IV se muestra los resultados de la investigación donde se ha evaluado la ejecución del sistema, los cuales son descritos en este capítulo. El sistema es de fácil manejo y de una interfaz amigable para todos aparte que también recomendara que hacer al momento de estar realizando las ventas y reportes.

CAPÍTULO I
REVISIÓN DE LITERATURA

1.1. Marco teórico

1.1.1. Sistema de información

Un sistema de información es un conjunto de elementos interrelacionados con el propósito de prestar atención a las demandas de información de una organización, para elevar el nivel de conocimientos un mejor apoyo en la toma de decisiones y desarrollo de acciones.(Carrizo & Rojas, 2018)

Un sistema de información realiza cuatro actividades básicas:

Entrada de información: Es el proceso mediante el cual el sistema de información toma los datos que requiere para procesar la información. Las entradas pueden ser manuales o automáticas. Las manuales son aquellas que se proporcionan en forma directa por el usuario, mientras tanto que las automáticas son datos o información que provienen o son tomados de otros sistemas o módulos.(Samaniego Pallaroso, 2012)

Almacenamiento de información: El almacenamiento es una de las actividades o capacidades más importantes que tiene una computadora, ya que a través de esta propiedad el sistema puede recordar la información guardada en la sección o proceso anterior. Esta información suele ser almacenada en estructuras de información denominadas archivos. La unidad típica de almacenamiento son los discos magnéticos o discos duros, los discos flexibles o diskettes y los discos compactos (CD-ROM).(Samaniego Pallaroso, 2012)

Procesamiento de información: Es la capacidad del sistema de información para efectuar cálculos de acuerdo con una secuencia de operaciones preestablecida. Estos cálculos pueden efectuarse con datos introducidos recientemente en el sistema o bien con datos que están almacenados. Esta característica de los sistemas permite la transformación de datos fuente de información que pueden ser utilizadas para la forma de decisiones lo que hace posible, entre otras cosas que un tomador de decisiones genere una proyección financiera a partir de los datos que contienen un estado de resultados o un balance general de un año base.

Salida de información: Es la capacidad de un sistema de información para sacar la información procesada o bien datos de entrada al exterior. Las unidades típicas de salida son las impresoras, terminales, cintas magnéticas, la voz y los plotters, entre otros. Es importante aclarar que la salida de un sistema de información puede constituir la entrada a otro sistema de información o modulo. En este caso, también existe una interface automática de salida.

Profesores de Administración de Empresas, un sistema de información es un organismo que recolecta, procesa, almacena y distribuye información. Son indispensables para ayudar a los gerentes a mantener ordenada su compañía, analizar nuevos productos que coloquen en un buen lugar a la organización.(Kendall Kenneth & Kendall Julie, 2005)

1.1.2. Sistema de consulta

Las consultas son el mecanismo mediante el cual el usuario final (humano o cibernético) accede al conocimiento y a los recursos.

Ingeniería de software

La ingeniería de software es el establecimiento que comprende todos los aspectos de la traducción del software y uso de principios robustos de la ingeniería a fin de obtener económicamente software que sea fiable y que funciones eficientemente sobre maquinas reales.(Aliaga Payehuanca, 2016)

Es la aplicación práctica del conocimiento científico en el diseño y construcción de programas de computadoras y la documentación asociada requerida para

desarrollar, operar y mantenerlos. Se conoce también como desarrollo de software o producción de software.(Carrizo & Rojas, 2018)

La ingeniería de software no solo comprende los procesos técnicos del desarrollo del software sino también con actividades tales como la gestión del proyecto y desarrollo de herramientas, métodos y teorías de apoyo a la producción de software.(Peña, 2006)

1.1.3. Software

Se conoce como software al equipamiento lógico o soporte lógico de una computadora digital; comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos del sistema, llamados hardware. Tales componentes lógicos incluyen, entre muchos otros, aplicaciones informáticas como el procesador de textos, que permite al usuario realizar todas las tareas concernientes a la edición de textos o el software de sistema tal como el sistema operativo, que, básicamente, permite al resto de los programas funcionar adecuadamente, facilitando la interacción con los componentes físicos y el resto de las aplicaciones, proporcionando también una interfaz para el usuario.(Peña, 2006)(EcuRed, 2017)

1.1.4. Programación orientada a objetos

Es un paradigma más de programación en el que un sistema se expresa como un conjunto de objetos que interactúan entre ellos. Un paradigma de programación nos proporciona una abstracción del sistema real a algo que podemos programar y ejecutar, y puede decirse que el tipo de abstracción está directamente relacionada con los problemas que puede resolver o al menos con la facilidad con que podremos resolverlos. Mientras que el lenguaje ensamblador es una abstracción del procesador, podríamos decir que otros lenguajes de programación como son abstracciones del propio lenguaje ensamblador. Aunque han supuesto un importante avance sobre éste, aún obligan a los programadores a pensar en términos de la estructura del ordenador en lugar de la estructura del problema que están intentando solucionar.(Ginesta & Peña González, 2005)

1.1.5. Análisis y Diseño de Sistemas Orientado a Objetos

Es un enfoque cuyo propósito es facilitar el desarrollo de sistemas que deben cambiar con rapidez en respuesta a entornos de negocios dinámicos. Es difícil trabajar bien con técnicas orientadas a objetos en situaciones en las cuales los sistemas de información complicados requieren mantenimiento, adaptación y rediseño de manera continua. Los enfoques orientados a objetos utilizan el estándar de la industria para la modelación de sistemas orientados a objetos, el lenguaje unificado de modelación [UML, Unified Modeling Languagé), para analizar un sistema en forma de modelo de casos de uso.(Gomez Cutipa, 2017)

La programación orientada a objetos difiere de la programación tradicional de procedimientos en que la primera examina los objetos que conforman un sistema. Cada objeto es una representación en computadora de alguna cosa o suceso real. Los objetos se representan y agrupan en clases, que son óptimas para su reutilización y mantenimiento. Una clase define el conjunto de atributos y comportamientos que comparten los objetos que ésta contiene. (Kendall Kenneth & Kendall Julie, 2005)

1.1.6. PhpMyAdmin

PhpMyAdmin es un programa de libre distribución en PHP, creado por una comunidad sin ánimo de lucro, que sólo trabaja en el proyecto por amor al arte. Es una herramienta muy completa que permite acceder a todas las funciones típicas de la base de datos MySQL a través de una interfaz Web muy intuitiva. La aplicación en si no es más que un conjunto de archivos escritos en PHP que podemos copiar en un directorio de nuestro servidor Web, de modo que, cuando accedemos a esos archivos, nos muestran unas páginas donde podemos encontrar las bases de datos a las que tenemos acceso en nuestro servidor de bases de datos y todas sus tablas. La herramienta nos permite crear tablas, insertar datos en las tablas existentes, navegar por los registros de las tablas, editarlos y borrarlos, borrar tablas y un largo etcétera, incluso ejecutar sentencias SQL y hacer un backup de la base de datos.(Cobo, Angel , Gomez, Patricia, 2005)

1.1.7. El Lenguaje de Modelado Unificado UML

El UML (Unified Modeling Language) tiene sus orígenes en la necesidad que se

habían generado en la industria para construir modelos orientados a objetos. UML es ante todo un lenguaje que se centra en representación gráfica de un sistema. Es un lenguaje visual estándar por medio de diversos elementos y procesos que interactúan de alguna forma con el software Después de tantas discusiones fue presentado el proyecto de UML a OMG quienes adoptaron a UML como estándar en la industria del software. (Kendall Kenneth & Kendall Julie, 2005).

1.1.8. Diagramas UML

Un diagrama es la representación gráfica de una colección de elementos con sus relaciones, ofreciendo así una vista del sistema a modelar. Para poder representar de forma correcta un sistema, el lenguaje presenta una amplia variedad de diagramas para así visualizar el sistema desde diversas perspectivas. (Kendall Kenneth & Kendall Julie, 2005)

1.1.8.1. Diagrama de Caso de Uso

Un diagrama de Casos de Uso muestra las distintas operaciones que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuario u otras aplicaciones) Es una herramienta esencial para la captura de requerimientos y para la planificación y control de un proyecto interactivo. Los casos de Uso se representan en el diagrama por una elipse que denota un requerimiento solucionando por el sistema. Cada caso de uso es una operación completa desarrollada por los actores y por el sistema en un diálogo. El conjunto de casos de uso representa la totalidad de operaciones desarrolladas por el sistema.(Ticahuanca, 2017)(EcuRed, 2017)

Elementos de Casos de Uso.

- ✓ Actor: Es un usuario del sistema, que necesita o usa alguno de los casos de uso. Un usuario puede jugar más de un rol. Un solo actor puede actuar en muchos casos de uso; recíprocamente, un caso de uso puede tener varios actores. Los actores no necesitan ser humanos pueden ser sistemas externos que necesitan alguna información del sistema actual. También se puede encontrar tres tipos de relaciones,

como son:

- ✓ Comunica: (comunicantes) Entre un actor y un caso de uso, denota la participación del actor en el caso de uso determinado.
- ✓ Usa: (use) Relación entre dos casos de uso, denota la inclusión del comportamiento de un escenario en otro. Se utiliza cuando se repite un caso de uso en dos o más casos de uso separados. Frecuentemente no hay actor asociado con el caso de uso común.
- ✓ Extiende: (extends) Relación entre dos casos, denota cuando un caso de uso es una especialización de otro. Se usa cuando se describe una variación sobre el normal comportamiento.(Booch, Rumbaugh, & Jacobson, 1999)(Ginesta & Peña González, 2005)

1.1.8.2. Diagrama de Clases

Un diagrama de clases o estructura estática de un modelo, las cosas que existen en términos de clase, su estructura interna y relaciones entre ellas, es decir las características de cada una de las clases, interfaces colaboraciones y relaciones de dependencia y generalización. Un diagrama de clases está compuesto por los siguientes elementos:

Clase: Una clase es un conjunto de objetos que comparten una estructura común y un comportamiento común.

Los atributos o características de las clases pueden ser de tres tipos, según el grado de comunicación y visibilidad de ellos con el entorno, estos son:

Públicos (+): indican que el atributo será visible tanto fuera como dentro de la clase, es decir, es accesible desde todos lados.

Privados (-): indican que el atributo solo será accesible desde dentro de la clase (solo sus métodos lo pueden acceder)

Protegidos (#) indica que el atributo no será accesible desde afuera de la clase, pero si podrá ser accesado por métodos de la clase.

Los métodos u operaciones de una clase son la forma en cómo esta

interactúa con su entorno, estos pueden tener las características:

Publico (+): indican que el método será visible tanto fuera como dentro de la clase, es decir, es accesible desde todos lados.

Existen cinco tipos de relaciones diferentes entre clases: dependencia, generalización, asociación, agregación y composición.

a. **Dependencia:** Es una relación de uso, es decir una clase usa a otra, que la necesita para su cometido. Se representa con una flecha discontinua que va desde la clase utilizadora a la clase utilizada. Con la dependencia se muestra que un cambio en la clase utilizada puede afectar el funcionamiento de la clase utilizadora, pero no al contrario.

b. **Generalización**: Es un relación entre un elemento más general (el padre) y elemento más específico (el hijo). El elemento más específico es totalmente consistente con el elemento más general y contiene la información adicional, también se define como la herencia, donde tenemos una o varias clases padre o superclase o madre, y una clase hija o subclase. Por ejemplo, un animal es un concepto más general que un gato, un perro o un pájaro. Inversamente, un gato es un concepto más específico que un animal.

c. **Agregación**: Es un tipo especial de asociación que representa una relación estructural entre las clases donde el llamado agregado indica el todo y el componente es una parte del mismo.

d. **Asociación**: Relación estructural que describe un conjunto de conexiones entre objetos de forma bidireccional.

e. **Composición:** Es un tipo de agregación donde la relación de posesión es tan fuerte como para marcar otro tipo de relación.

Tabla 1
Relación entre clases

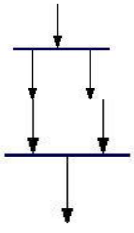
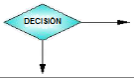
Nombre	Símbolo
Asociación	
Agregación	
Composición	
Dependencia	
Generalización	

Fuente: Ginesta & Peña González, 2005

1.1.8.3. Diagrama de actividades

Permiten modelar el comportamiento de un sistema o alguno de sus elementos, mostrando la secuencia de actividades o pasos que tienen lugar para la obtención de un resultado o la consecución de un determinado objetivo se detalla en la siguiente tabla:

Tabla 2
Elementos de diagramas de despliegue

Nombre	**Símbolo**	**Descripción**
Acción		Nodo de actividad primitiva ejecutable de asignación o computación.
Nodo de inicio		Nodo de control que indica el inicio de un flujo de control cuando una actividad es invocada.
Nodo fin de actividad		Nodo de control que indica el fin de todos los flujos dentro de una muestra el fin de la actividad.
Flujo de control		Eje de actividades para flujo de control conecta dos acciones. Usado para indicar secuencia.
Nodo de sincronización (fork)		Nodo de control que divide un flujo en dos o más flujos concurrentes (paralelos).
Nodo de concurrencia (join)		Nodo de control que sincroniza múltiples flujos.
Nodo de decisión		Nodo de control que selecciona entre dos o más flujos de salida.

Fuente: Ginesta & Peña González, 2005

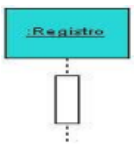
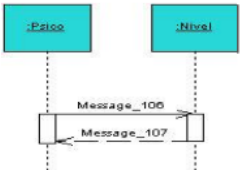
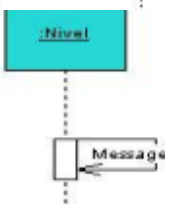
1.1.8.4. Diagrama de interacción

Estos son modelos que describen como los grupos de objetos que colaboran en algunos ambientes. Por lo general, un diagrama de interacción captura el comportamiento de un único caso de uso. Hay dos tipos de diagramas de interacción. “diagramas de secuencia y diagramas de colaboración”.(Kendall Kenneth & Kendall Julie, 2005)

Diagrama de secuencia

Un diagrama de secuencia es un tipo de diagrama de interacción en el cual se destaca el tiempo: los mensajes entre objetos vienen ordenados explícitamente por el instante en que se envían. Consta de dos ejes. Generalmente, el eje vertical es el eje del tiempo, transcurriendo éste de arriba a abajo. En el otro eje se muestran los objetos que participan en la interacción, siendo el primero de ellos el actor que inicia la ejecución de la secuencia modelada.(Carrera Jimenez, 2011), de cada objeto parte una línea discontinua, llamada línea de la vida como se muestra en la Tabla 3

Tabla 3
Elementos de diagrama de secuencia

Nombre	Símbolo	Descripción
Línea de vida		Indica el periodo en que estuvo vivo el objeto durante la secuencia de actividad.
Activación		Muestra el periodo de tiempo en el cual el objeto se encuentra desarrollando alguna operación.
Mensaje de un objeto a otro		Él envió de mensajes entre objetos se denota mediante una línea solida dirigida, desde el objeto que emite el mensaje hacia el objeto que lo ejecuta.
Mensaje a un mismo objeto		Como su nombre lo indica, es el mensaje que un objeto se envió a sí mismo.

Fuentes: Kendall Kenneth & Kendall Julie, 2005

1.1.8.5. Diagramas de Colaboración

Es una forma de representar interacción entre los objetos, es decir, las relaciones entre ellos y la secuencia de los mensajes de las iteraciones que están indicadas por un número, a diferencia de los diagramas de secuencia, pueden mostrar el contexto de la operación (cuáles objetos son atributos, cuáles temporales) y ciclos en la ejecución. Muestra como varios objetos colaboran en un solo caso de uso.

1.1.8.6. Diagrama de Estados

Muestra el conjunto de estado por los cuales pasa un objeto durante su vida en una aplicación con los cambios que permiten pasar de un estado a otro.

Está representado principalmente por los siguientes elementos. “Estado, elemento y transición”.

Estado: Identifica un período de tiempo del objeto (no instantáneo) en el cual el objeto está esperando alguna operación, tiene cierto estado característico o puede recibir cierto tipo de estímulos.(Ginesta & Peña González, 2005)

1.1.8.7. Diagramas de Implantación

Muestran aspectos de la implementación del sistema, donde se incluyen la estructura del código fuente y su implementación en tiempo real con la estructura física del sistema. Hay dos tipos de diagramas de implementación:

1.1.8.8. Diagrama de componentes

Un diagrama de componentes muestra dependencias entre los componentes. Cada componente ofrece algunas interfaces y utiliza otras. Si las dependencias entre componentes se hacen a través de interfaces, los componentes se pueden sustituir por otros componentes que realicen las mismas interfaces. (Booch et al., 1999)

Diagramas de despliegue

Son aquellos que muestran las relaciones físicas entre los componentes de software y hardware en el sistema desarrollado, es decir cómo se encuentran y se mueven los componentes y los objetos. En otras palabras, los diagramas de despliegue muestran la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes de software, procesos y objetos que residen en ellos.

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema mostrando la configuración de los elementos de hardware y mostrando cómo los elementos y artefactos del software se trazan en esos nodos.(Alvaro PEÑA, 2005)(Booch et al., 1999)

Tabla 4
Elementos del diagrama de despliegue

Nombre	Símbolo	Descripción
Nodo		Es un objeto físico en tiempo de ejecución que representa un recurso computacional, generalmente con memoria y capacidad de procesamiento.
Componente		Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas
Interface		Las interfaces se utilizan como lazo de unión entre unos componentes y otros.

Fuente: Kendall Kenneth & Kendall Julie, 2005

1.1.9. Software Libre

El «software libre» es una cuestión de libertad, no de precio. Para comprender este concepto, debemos pensar en la acepción de libre como en «libertad de expresión» y no como en «barra libre de cerveza». Con software libre nos referimos a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. Nos referimos especialmente a cuatro clases de libertad para los usuarios de software:(Stallman, 2004)

a. **Libertad 0:** la libertad para ejecutar el programa sea cual sea nuestro propósito.

b. **Libertad 1:** la libertad para estudiar el funcionamiento del programa y adaptarlo a tus necesidades —el acceso al código fuente es condición indispensable para esto.

c. **Libertad 2:** la libertad para redistribuir copias y ayudar así a tu vecino.

d. **Libertad 3:** la libertad para mejorar el programa y luego publicarlo para el bien de toda la comunidad —el acceso al código fuente es condición indispensable para esto.

Software libre es cualquier programa cuyos usuarios gocen de estas libertades. De modo que deberías ser libre de redistribuir copias con o sin modificaciones, de forma gratuita o cobrando por su distribución, a cualquiera y en cualquier lugar. Gozar de esta libertad significa, entre otras cosas, no tener que pedir permiso ni pagar para ello. Asimismo, deberías ser libre para introducir modificaciones y utilizarlas de forma privada, ya sea en tu trabajo o en tu tiempo libre, sin siquiera tener que mencionar su existencia. Si decidieras publicar estos cambios, no deberías estar obligado a notificarlo de ninguna forma ni a nadie en particular. La libertad para utilizar un programa significa que cualquier individuo u organización podrán ejecutarlo desde cualquier sistema informático, con cualquier fin y sin la obligación de comunicárselo subsiguientemente ni al desarrollador ni a ninguna entidad en concreto.(Stallman, 2004)

1.1.9.1. Desarrollo de Software Libre

Las condiciones de licenciamiento de los programas libres permiten la construcción comunitaria de software. Los desarrolladores de software pueden acudir a inmensas colecciones de programas y bibliotecas altamente funcionales e intensamente probadas. Esto reduce el esfuerzo y el riesgo de desarrollo, comparado con la alternativa de empezar de cero. (Stallman, 2004)

1.1.10. Lenguajes de Programación

Un lenguaje de programación se refiere a cualquier lenguaje artificial que pueda ser empleado para definir una secuencia de instrucciones para su procesamiento por una computadora u ordenador. Por lo general, se encuentra formado por un conjunto de símbolos y reglas de tipo semánticas y sintácticas, que permiten a los programadores definir de manera precisa acerca de qué datos debe operar una computadora, cómo estos datos deben ser almacenados o transmitidos y qué acciones debe tomar ante diferentes eventos.(EcuRed, 2017)

- **HTML**

  Utiliza marcas para describir la forma en la que deberían aparecer el texto y los gráficos en un Navegador web que, a su vez, están preparados para leer esas marcas y mostrar la información en un formato estándar. Algunos navegadores Web incluyen marcas adicionales que sólo pueden leer y utilizar ellos. La utilización de marcas no estándares en lugares de especial importancia no es recomendable.(EcuRed, 2017)

- **PHP**

  Lenguaje de programación, interpretado, diseñado originalmente para la creación de Páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+. PHP es un Acrónimo recursivo que significa PHP Hypertext Pre-processor (inicialmente PHP

Tools, o, Personal Home Page Tools). Fue creado originalmente por Rasmus Lerdorf en 1994; sin embargo la implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la PHP License, la Free Software Foundation considera como software libre.(EcuRed, 2017)

- **XAMPP**

  Es un servidor independiente de plataforma de código libre, te permite instalar de forma sencilla Apache en tu propio ordenador sin importar tu sistema operativo y lo mejor es de uso gratuito XAMPP incluye servidores de bases de datos como MySQL con sus respectivos gestores phpMyAdmin y phpSQLiteAdmin, etc, entre muchas cosas más. También es una herramienta de desarrollo que te permite probar tu trabajo (páginas web o programa por ejemplo) en tu propio ordenador sin necesidad de tener que acceder a internet. (Ben, Laurie., 2005)

- **Sublime text**

  Es un editor de código multiplataforma, ligero y con pocas concesiones a las florituras. Es una herramienta concebida para programar sin distracciones. Su interfaz de color oscuro y la riqueza de coloreado de la sintaxis, centra nuestra atención completamente. Permite tener varios documentos abiertos mediante pestañas, e incluso emplear varios paneles para aquellos que utilicen más de un monitor. Dispone de modo de pantalla completa, para aprovechar al máximo el espacio visual disponible de la pantalla.(EcuRed, 2017)

- **CakePHP**

  Es un framework de desarrollo de aplicaciones web escrito en PHP, creado sobre los conceptos de Ruby on Rails.

  CakePHP incluye las clases Controlador (Controller), Modelo (Model) y Vista (View), pero también incluye otras clases y objetos que hacen que el desarrollo en MVC sea un poco más rápido y agradable.(EcuRed, 2017)

La aplicación está en tres partes principales:

- La capa MODELO
- La capa VISTA
- La capa CONTROLADOR

MVC: Porque es un patrón de diseño de software probado y se sabe que funciona. Con MVC la aplicación se puede desarrollar rápidamente, de forma modular y mantenible. Separar las funciones de la aplicación en modelos, vistas y controladores hace que la aplicación sea muy ligera. Estas características nuevas se añaden fácilmente y las antiguas toman automáticamente una forma nueva.

El diseño modular permite a los diseñadores y a los desarrolladores trabajar conjuntamente, así como realizar rápidamente el prototipo. Esta separación también permite hacer cambios en una parte de la aplicación sin que las demás se vean afectadas.

Entendiendo Modelo-Vista-Controlador: Las aplicaciones CakePHP bien escritas siguen el patrón de diseño de software MVC (Modelo-Vista-Controlador). Programar utilizando MVC consiste en separar la aplicación en tres partes principales. El modelo representa los datos de la aplicación, la vista hace una presentación del modelo de datos, y el controlador maneja y enrruta las peticiones [requests] hechas por los usuarios.

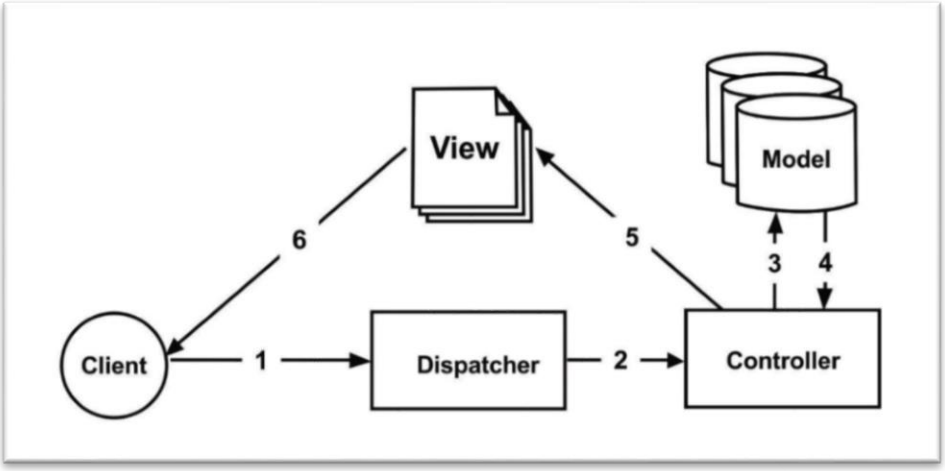


Figura 1. Petición modelo vista controlador básico

Estructura de archivos de CakePHP

Tras descargar y extraer CakePHP, estos serán los ficheros y carpetas que deberías ver:

- App
- Cake
- Vendors
- Htaccess
- index.php
- README

Observarás 3 carpetas principales:

- **La carpeta app:** Será donde haremos nuestra magia: es donde se ubicarán los ficheros de tu aplicación.
- **La carpeta cake:** Es donde nosotros hemos hecho nuestra magia. Comprométete a no modificar los ficheros de esta carpeta. No podremos ayudarte si has modificado el núcleo.
- **La carpeta vendors:** Es donde ubicarás las librerías PHP de terceros que necesites.

1.1.11. Sistematización

La sistematización se entenderá en esta guía como la organización y ordenamiento de la información existente con el objetivo de explicar los cambios (+ ó -) sucedidos durante un proyecto, los factores que intervinieron, los resultados y las lecciones aprendidas que dejó el proceso. El objetivo de un proceso de sistematización es facilitar que los actores de los procesos de desarrollo se involucren en procesos de aprendizaje y de generación de nuevos conocimientos o ideas de proyectos e iniciativas de políticas/estrategias a partir de las experiencias documentadas, datos e informaciones anteriormente dispersos. Los procesos de sistematización permiten:

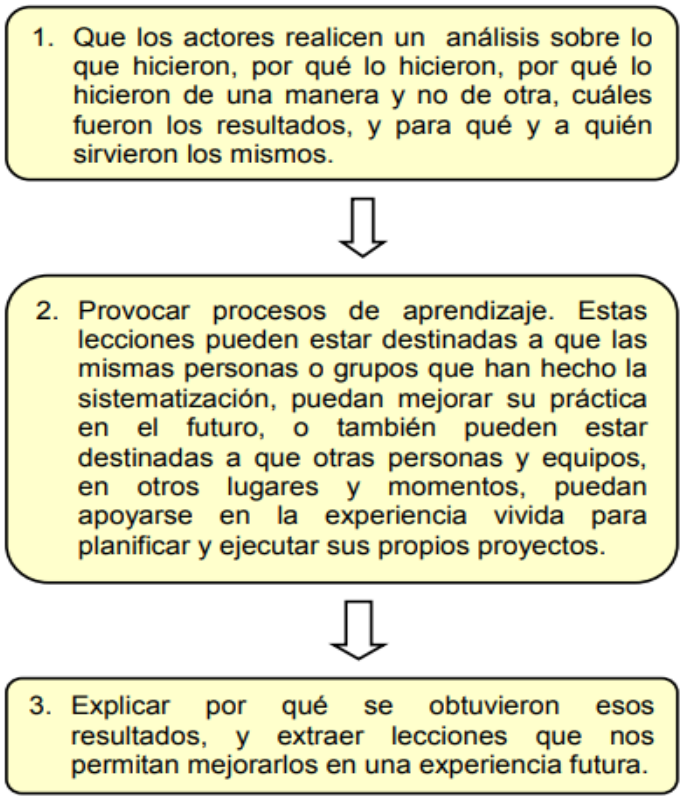


Figura 2. Esquema básico de sistematización

Los resultados de una experiencia son fundamentales, y describirlos es parte importante de toda sistematización, pero lo que más interesa en el proceso de sistematización.(Acosta, 2005)

1.1.12. Base de datos

Base de Datos o Banco de Datos (BB.DD.). Es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido, una Biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta.(EcuRed, 2017)

1.1.13. Metodología XP (Extreme Programming)

Extreme Programming o Programación eXtrema, es una de las metodologías llamadas "ágiles", para el desarrollo de proyectos de software. Se basa en los principios de la simplicidad, la comunicación, la retroalimentación y el coraje para involucrar a todo el equipo (y a los usuarios o clientes) en la gestión del proyecto. En 1996, Kent Back y Ward Cunningham pusieron en práctica una nueva metodología primando la simplicidad y evitando los hábitos que convertían las cosas fáciles en difíciles durante el desarrollo de un proyecto en DaimlerChrysler. El resultado fue la metodología eXtreme Programming o XP (que nada tiene que ver con Microsoft). (Ginesta & Peña González, 2005)

1.1.13.1. Roles de la metodología XP:

a. PROGRAMADOR:

Es el Responsable de implementar las historias de usuario por el cliente. Además, estima el tiempo de desarrollo de cada historia de usuario para que el cliente pueda asignarle prioridad dentro de la iteración. Cada iteración incorpora nueva funcionalidad de acuerdo a las prioridades establecidas por el cliente. El Programador también es responsable de diseñar y ejecutar los test de unidad del código que ha implementado o modificado.

b. CLIENTE:

Determina la funcionalidad que se pretende en cada iteración y define las prioridades de implementación según el valor de negocio que aporta cada historia. El Cliente también es responsable de diseñar y ejecutar los test de aceptación.

c. ENCARGADO DE PRUEBAS (TESTER):

Es el Encargado de ejecutar las pruebas regularmente, difunde los resultados dentro del equipo y es también el responsable de las herramientas de soporte para pruebas.

d. ENCARGADO DE SEGUIMIENTO (TRACKER):

Una de las tareas más importante del tracker, consiste en seguir la evolución de las estimaciones realizadas por los programadores y compararlas con el tiempo real de desarrollo. De esta forma, puede brindar información estadística en lo que refiere a la calidad de las estimaciones para que puedan ser mejoradas.

e. ENTRENADOR (COACH):

Es Responsable del proceso en general. Se encarga de iniciar y de guiar a las personas del equipo en poner en marcha cada una de las prácticas de la metodología XP.

f. CONSULTOR:

Es un Miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo para resolver un problema específico.

g. GESTOR (BIG BOSS):

Es el vínculo entre el cliente y programadores. Experto en tecnología y labores de gestión. Construye el plantel del equipo, obtiene los recursos necesarios y maneja los problemas que se generan. Administra las reuniones, su labor fundamental es de coordinación.(Meléndez Valladarez, Gaitan, & Pérez Reyes, 2016)

1.1.13.2. Fases de la programación extrema

La Programación Extrema consta de 4 fases, las cuales son:(Meléndez Valladarez et al., 2016)

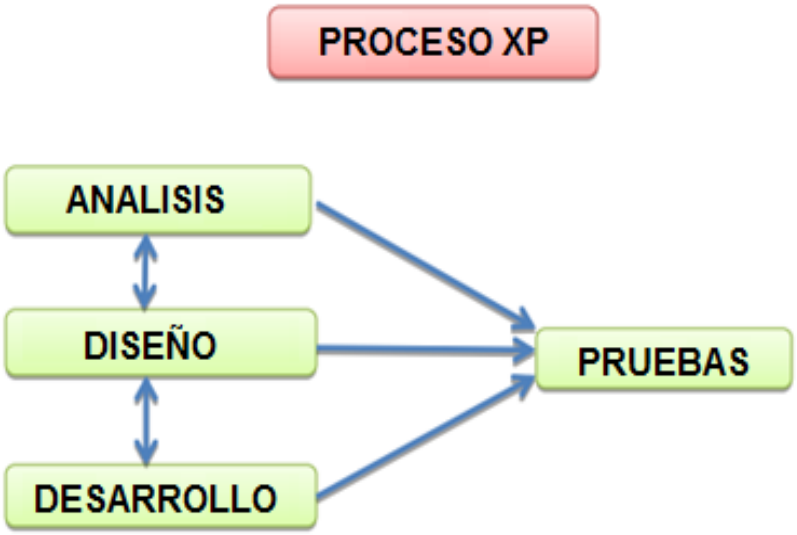


Figura 3. Fases de la Programación – XP

A. ANALISIS:

La metodología XP plantea en análisis como un permanente diálogo entre la parte empresarial y técnica del proyecto, en la que los primeros decidirán el alcance ¿Qué es lo realmente necesario del proyecto?, la prioridad qué debe ser hecho en primer lugar, la composición de las versiones que debería incluir. (Laura Murillo, 2014)

B. DISEÑO:

El Propósito del diseño es de crear una arquitectura para la naciente implementación, el diseño arquitectural sólo puede comenzar una vez que el equipo tenga un entendimiento razonable de los requerimientos del sistema. El diseño, como el análisis, nunca termina realmente hasta que el sistema final es entregado.(Laura Murillo, 2014)

C. DESARROLLO:

Esta etapa debe reunir las siguientes características o cualidades.(Laura Murillo, 2014)

- El cliente está siempre disponible
- Se debe escribir código de acuerdo a los estándares
- Desarrollar la unidad de pruebas primero
- Todo el código debe programarse por parejas
- Integrar frecuentemente
- Todo el código es común a todos

D. PRUEBAS:

Todo el código debe ir acompañando, Los casos de prueba se escriben antes que el código. Los desarrolladores escriben pruebas unitarias y los clientes especifican pruebas funcionales.(Laura Murillo, 2014)

1.1.14. Estándar ISO/IEC 9126

Esta norma internacional fue publicada en 1992, la cual es usada para la evaluación de la calidad de software, La norma ISO/IEC 9126 permite especificar y evaluar la calidad del software desde diferentes criterios asociados con adquisición, requerimientos, desarrollo, uso, evaluación, soporte, mantenimiento, aseguramiento de la calidad y auditoria de software.

El objetivo de esta norma ISO/IEC 9126 es proporcionar un modelo de calidad que sirve como elemento central en un proceso de evaluación. El modelo de calidad que propone la norma puede aplicarse a cualquier tipo de software incluido el desarrollo para el ámbito educativo.(Abud Figueroa, 2012)

Figura 4. Norma de evaluación ISO/IEC 9126

1.1.14.1. Características generales:

a. Funcionalidad

Es la capacidad de un producto software para satisfacer los requisitos funcionales y las necesidades de los usuarios. Se divide en 5 criterios:

Adecuación: La capacidad del software para proveer un adecuado conjunto de funciones que cumplan las tareas y objetivos especificados por el usuario.

Exactitud: Evalúa el resultado final que obtiene el software y si tiene consistencia a lo que espera de él.

Interoperabilidad: consiste en revisar si el sistema puede interactuar con otro sistema independiente.

Seguridad: verifica si el sistema puede impedir el acceso a personal no autorizado.

Conformidad de la funcionalidad: La capacidad del software de cumplir los estándares referentes a la funcionalidad.

b. Confiabilidad

Es la capacidad de un producto software para mantener su nivel de desempeño bajo condiciones establecidas, por un periodo de tiempo. Se divide en 4 criterios:

Madurez: Se debe verificar las fallas del sistema y si muchas de estas han sido eliminadas durante el tiempo de pruebas o uso del sistema.

Tolerancia a errores: Evalúa si la aplicación desarrollada es capaz de manejar errores.

Recuperabilidad: Verificar si el software puede reasumir el funcionamiento y restaurar datos perdidos después de un fallo ocasional.

Conformidad de la fiabilidad: La capacidad del software de cumplir a los estándares o normas relacionadas a la fiabilidad.

c. Usabilidad

Es la capacidad del software de ser entendido, aprendido, y usado en forma fácil y atractiva. Algunos criterios de funcionalidad, fiabilidad y eficiencia afectan la usabilidad, pero para los propósitos de la ISO/IEC 9126 ellos no clasifican como usabilidad. La usabilidad está determinada por los usuarios finales y los usuarios indirectos del software, dirigidos a todos los ambientes, a la preparación del uso y el resultado obtenido.se divide en 5 criterios:

Entendimiento: La capacidad que tiene el software para permitir al usuario entender si es adecuado, y de manera fácil como ser utilizado para las tareas y las condiciones particulares de la aplicación.

Aprendizaje: Determina que tan fácil es para el usuario aprender a utilizar el sistema.

Operabilidad: La manera como el software permite al usuario operarlo y controlarlo.

Atracción: Verifica que tan atractiva se ve la interfaz de la aplicación.

Conformidad de uso: La capacidad del software de cumplir los estándares o normas relacionadas a su usabilidad.

d. Eficiencia

Es la capacidad de un producto software de proporcionar un rendimiento apropiado, de acuerdo a la cantidad de recursos usados bajo condiciones establecidas.se divide en 3 criterios:

Comportamiento de tiempos: Verifica la rapidez en que responde el sistema.

Comportamiento de recursos: Determina si el sistema utiliza los recursos de manera eficiente.

Conformidad de eficiencia: La capacidad que tiene el software para cumplir con los estándares relacionados a la eficiencia.

e. Mantenimiento

Es la capacidad de un producto software para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptación del software a cambios en el entorno, las especificaciones funcionales se divide en 5 criterios:

Estabilidad: Verifica si el sistema puede mantener su funcionamiento a pesar de realizar cambios.

Facilidad de analizado: Determina si la estructura de desarrollo es funcional con el objetivo de diagnosticar fácilmente las fallas.

Facilidad de Cambios: Verifica si el sistema fácilmente modificado.

Facilidad de prueba: Evalúa si el sistema puede ser probado fácilmente.

Conformidad de facilidad de mantenimiento: La capacidad que tiene el software para cumplir con los estándares de facilidad de mantenimiento.

f. Portabilidad

La capacidad de un producto software de ser transferido de un ambiente a otro. Nota: El ambiente puede ser organizacional, de software o de hardware. Se divide en 5 criterios:

Adaptabilidad: Es como el software se adapta a diferentes entornos especificados (hardware o sistemas operativos) sin que implique reacciones negativas ante el cambio. Incluye la escalabilidad de capacidad interna (Ejemplo: Campos en pantalla, tablas, transacciones, formatos de reporte, etc.).

Facilidad de instalación: Verifica si el software se puede instalar fácilmente

Coexistencia: La capacidad que tiene el software para coexistir con otro o varios software.

Capacidad de reemplazamiento: Determina la facilidad con la que el software puede remplazar otro software similar.

Conformidad de portabilidad: La capacidad que tiene el software para cumplir con los estándares relacionados a la portabilidad.(Abud Figueroa, 2012)

1.2. Definición de términos básicos

Ventas: Venta es una acción que se genera de vender un bien o servicio a cambio de dinero. Las ventas pueden ser por vía personal, por correo, por teléfono, entre otros medios. El término venta es de origen latín "vendita", participio pasado de "venderé".

Arquitectura: Una arquitectura es un entramado de componentes funcionales, aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informáticos, de manera que puedan ser utilizadas eficazmente dentro de la organización.

Automatización: Es la realización de una tarea del hombre con la ayuda de una computadora diseñada y acondicionada para una tarea especificada

Proceso: un conjunto de actividades, material y/o flujo de información que transforma un conjunto de entradas en resultados definidos.

Archivo: Es un conjunto de Registros homogéneos (de un mismo tipo).

Sistema: Es el conjunto de elementos relacionados entre sí con un propósito determinado (único). Es un conjunto u ordenación de elementos organizados para llevar a cabo algún método, procedimiento o control mediante el procesamiento de información.

Sistema de información: Es un conjunto de procedimientos organizados que, cuando se ejecutan, proporcionan información para la toma de decisiones y/o el control de la organización. Es un conjunto de personas, datos y procedimientos que funcionan en conjunto; el énfasis en sistemas significa que los variados componentes buscan un objetivo común para apoyar las actividades de la organización.

Lenguaje de programación: Es cualquier lenguaje según el cual se expresan o se escriben las instrucciones (programas) de procesamiento de la computadora. Lenguajes en el cual se escriben las instrucciones que controlan el movimiento y procesado de datos.

Programación: Conjunto de instrucciones escritas en un lenguaje particular, que representa la resolución del problema. En otros términos se puede decir que un "programa" es la elaboración de un algoritmo efectuado en lenguaje para ordenador.

Prototipo: Un Prototipo es un modelo (representación, demostración o simulación) fácilmente ampliable y modificable de un sistema planificado, probablemente incluyendo su interfaz y su funcionalidad de entradas y salidas.

Enrutador: Un enrutador es un dispositivo que regula el tráfico en la Internet haciéndolo más eficiente para cada paquete. Un paquete puede pasar a través de muchos enrutadores antes de llegar a su destino.

Tiempo: Sucesión de fenómenos que va irreversiblemente del pasado al futuro.

1.3. Operacionalización de variables

Tabla 5
Operacionalización de variables

VARIABLE	DIMENSIÓN	INDICADORES	INDICE
DEPENDIENTE	Software	Servicial	Si
Sistema de ventas			No
INDEPENDIENTE	Procesos de ventas	Toma de decisiones	Efectiva
ventas			Mala
		Reducción de tiempo en las ventas	Buena
			Mala
			Regular
		Reportes	Optimo
			Deficiente

Fuente: Datos e información del trabajo de investigación

1.4. Antecedentes

La implementación de sistemas de ventas avanzados están siendo seguida y desarrollada por muchos investigadores, sino también por los diferentes gobiernos locales, medios de comunicación y el estado, por esta razón serán mencionados algunos estudios realizados en diferentes países del mundo, en el Perú, la región y especialmente en nuestra ciudad.

Según Mayta (2018) Cuyo objetivo principal fue desarrollar un Sistema de Información que contribuya eficientemente en el comercio electrónico de equipos y accesorios de celulares en la ciudad de Juliaca. Este trabajo de investigación llegó a la conclusión de que el desarrollo de un Sistema de Información si aumenta las ventas de equipos y accesorios de celulares en Juliaca - 2018.

Gomez (2017), En su investigación señala en su objetivo general: Desarrollar un sistema de ventas para micro y pequeñas empresas, aplicado a la empresa San Santiago, acora-puno – 2017. La metodología fue XP (programación extrema), y su conclusión fue: Se mejoró SIGNIFICATIVAMENTE el proceso de ventas en la micro y pequeña empresa San Santiago

Ticahuanca (2017) Sistema de gestión comercial aplicando ERP en la empresa Grupo Perusis S.A.C. El estudio fue realizado a la empresa Grupo Perusis, la cual se dedica a brindar soluciones tecnológicas. Al implementar un sistema ERP que comunique sus procesos principales mejoro el manejo de información enormemente.

Ramos (2017) Tiene como objetivo desarrollar e implementar un sistema eficiente de gestión de libros contables que permiten llegar a obtener información eficaz y oportuna, Con el desarrollo e implementación del sistema de gestión de libros contables se agilizo eficientemente los procesos contables en la empresa Constructora y Servicios HNOS M&R de Puno.

Aliaga (2017) El objetivo general es: Implementar una plataforma de software para la verificación y detección de plagio en documentos de investigación con un corpus de referencia en la Universidad Nacional del Altiplano – Puno y concluye análisis de los procedimientos que se realizan en la actualidad y haciendo uso de un software con un corpus de referencia para determinar si en los trabajos de investigación de la universidad nacional del altiplano existe o no plagio

Vilcapaza y Vilca (2016) El objetivo principal de la presente investigación es de Optimizar el registro genealógico de cuyes (Cavia Porcellus Linnaeus) en el I.E.S.T.P. (Instituto de Educación Superior Tecnológico Publico) Pedro Vilcapaza de la Provincia de Azángaro, mediante la implementación de un sistema de información;

usando tecnología Web, Los resultados fueron la automatizacion el registro de cuyes (Cavia porcellus Linnaeus) en la I.E.S.T.P. Pedro Vilcapaza de la Provincia de Azángaro2016.

Chura (2015) Se ha implementado un Sistema de administración de ventas de una micro y pequeña empresa en Azángaro, 2015. Se concluye que el sistema de administración de ventas que se implantó, mejoró el proceso de ventas de las variables que brinda la información de cómo se realizaban las ventas.

Galindo (2012), Su objetivo general: Implementar un sistema de información web orientado a la gestión educativa de un centro educativo especial que brinde soporte a las labores y actividades pedagógicas efectuadas por los especialistas de esta institución. Concluye que comprueba la capacidad de integración de aplicaciones construidas bajo la plataforma NET Framework con proyectos de código abierto logrando una significativa reducción de costos.

Mendoza (2011) En su investigación su objetivo general señala: Implementar un sistema automatizado que optimice la gestión de los procesos administrativos del área de servicios médicos. Concluye que la comunicación con el cliente represento una clave fundamental para poder validar los requisitos y cumplir con sus necesidades o requerimientos que se da a partir de cada una de las iteraciones a lo largo del proceso de desarrollo, facilitando la labor de muchas tareas e impactando de manera positiva en el tiempo como análisis y diseño.

Jiménez (2009), En su investigación señala en su objetivo general: Desarrollar una aplicación vía intranet que permite el trámite de documentos de pago a proveedores. La metodología aplicada son las fases análisis y diseño tiene como uno de sus propósito mejorar el proceso de trámite de los documentos de pago a proveedores y la implementación del sistema con el fin de apoyar las labores administrativas de una institución organizada en unidades como la PUCP.

Chávez (2010), El objetivo principal de esta investigación fue: analizar, diseñar, desarrollar e implementar un Sistema de Información para el Control, Seguimiento y Mantenimiento del Equipamiento Hospitalario en el Hospital Central de la Fuerza

Aérea del Perú. Se llegó a la siguiente conclusión: El mantenimiento es considerado hoy en día un factor estratégico, por ello que el Hospital Central de la FAP aspira a ser más competitivo y eficiente, adoptando técnicas y sistemas que le permitan tener organizada y actualizada esa gran cantidad de información para llevar a cabo una buena gestión del mismo.

Mosquera (2007), Su objetivo es: Realizar el análisis, diseño e implementación de un sistema integral de gestión hospitalaria que permita la administración de la información para centros de salud públicos, el mantenimiento de la información consistente, relacionada y centralizada para lograr la sinergia en los procesos, la implantación de una arquitectura que soporte los escalamientos de los sistemas de información. Y se llegó a la siguiente conclusión: A través del sistema se canaliza la información mediante una sola vía de ingreso, centralizando de esta manera toda la información ingresada. Apoyándose en esto se gestionará la información para realizar una toma de decisiones real y precisa.

Esteban (2011), Objetivo: Desarrollar un prototipo de Sistema Experto que contenga el conjunto de procesos de razonamiento y conocimiento requeridos por un experto en selección de personal, utilizando lenguajes de programación declarativa de libre distribución y que sirva de apoyo para la toma de decisiones. Y concluye que La Selección de Personal es claramente uno de los procesos más importantes y críticos para las organizaciones, aunque tiene un alto nivel de incertidumbre y subjetividad dependiendo del contexto donde esté siendo ejecutado. Se espera que el sistema desarrollado mediante este trabajo pueda soportar el proceso en las organizaciones.

CAPÍTULO II
PLANTEAMIENTO DEL PROBLEMA

2.1. Identificación del problema

2.1.1. Planteamiento del problema

En la actualidad la competencia ha crecido bastante en el rubro de ventas por lo cual la mayoría de empresas están optando por la implementación de un sistema avanzado para que puedan controlar las ventas y/o tomar decisiones en beneficio de la generación de más ganancias para empresa. Además con el avance de la tecnología la implementación de un sistema avanzado no es muy costoso como lo era años atrás.

En la actualidad la planta de criadero de truchas Arapa S.A.C., no cuenta con un sistema de ventas, que permita un óptimo manejo de información en la empresa. Los sistemas avanzados son de gran ayuda para las empresas en todos los rubros que existen en el mercado porque son independientes y toman sus propias decisiones en favor de la empresa.

Científicos de la Universidad Carlos III de Madrid (UC3M) estudian cómo mejorar el desarrollo de sistemas de computación avanzados para conseguir software más rápido en el marco de RePhrase, un nuevo proyecto de investigación del programa Horizonte 2020 de la Unión Europea. Estas nuevas técnicas permitirán mejorar aplicaciones como procesos de fabricación industrial, seguimiento del tráfico ferroviario, así como el diagnóstico de enfermedades mentales.

2.2. Justificación

El presente proyecto de investigación es de gran interés que responde a la necesidad de contar con un sistema de ventas avanzado para la planta de criadero de truchas Arapa S.A.C. del distrito de Arapa puesto que es una herramienta muy útil para la empresa.

Por otro lado, al implementar nuestro sistema de ventas avanzado aparte de solo controlar las ventas también ayudo al vendedor en la tomar decisiones al momento de realizar ventas para que así pueda haber generar más ganancias en la empresa ya que el sistema dará un aviso cuando el producto a vender se ese agotando y dará una sugerencia del producto que haya más, por lo que reducimos así la complejidad de tomar decisiones al momento de realizar las ventas.

Para el desarrollo, se utilizó la metodología de Programación Extrema para un proceso iterativo durante el análisis, implementación y pruebas de ejecución.

Esto nos condujo a la pregunta de investigación:

¿El desarrollo de un sistema mejorará el proceso de ventas en la planta de criadero de truchas Arapa S.A.C., del distrito de Arapa provincia de Azángaro Región Puno?

2.2.1. Delimitación de la investigación

El proyecto del desarrollo del sistema de ventas avanzado será de servidor local hasta un total de 10 máquinas ya que en la empresa el internet es muy lento debido a que solo se utiliza modem.

2.3. Objetivos

2.3.1. Objetivo general

Desarrollar un sistema de ventas de truchas para la planta de criaderos de truchas Arapa S.A.C. del distrito de Arapa Provincia de Azángaro, Región de Puno.

2.3.2. Objetivos específicos

- Analizar las ventas y/o stock para la toma de decisiones (Algoritmos avanzados).
- Analizar el tiempo en la eficiencia en la ventas y atención al cliente antes y después de la implementación del sistema
- Diseñar e Implementar un sistema informático de ventas en la planta de criadero de truchas Arapa S.A.C. del distrito de Arapa Provincia de Azángaro, Región de Puno utilizando la metodología Extrema – XP porque es la que más se adecua al presente trabajo.

2.4. Hipótesis

La implementación de un sistema avanzado mejorara el proceso de ventas de la planta de criadero de truchas Arapa S.A.C. del distrito de Arapa Provincia de Azángaro, Región de Puno.

CAPÍTULO III
MATERIALES Y MÉTODOS

3.1. Lugar de estudio

El presente trabajo se realizó en el Distrito de Arapa, Provincia de Azángaro Región Puno, en la comunidad de Escayapi está la planta de producción de truchas denominada Arapa S.A.C. que se encuentra ubicado a 3815 m.s.n.m., situado a 12° de latitud sur de la región Puno el cual se muestra en la Figura 5

Figura 5. Localización de la empresa

3.2. Tipo de investigación

La presente investigación corresponde al tipo de investigación demostrativa donde se trabaja con los productos para realizar las ventas en un sistema.

3.3. Ámbito o lugar de estudio

La investigación se desarrolló en la ciudad de Puno en los laboratorios de la maestría en Informática durante el periodo de estudios Enero – Abril del 2019

3.4. Población y muestra

3.4.1. Población

La población está constituida por los trabajadores de la empresa San Pedro San Pablo con un total de 10 trabajadores.

- 1 Administrador
- 3 Vendedores
- 4 Almaceneros
- 2 cajeros

3.4.2. Muestra

Tipo de muestreo a criterio del investigador, la muestra está conformada por la misma cantidad de personas de la población (10 trabajadores).

3.5. Metodología de desarrollo

La información para el presente estudio de investigación fue obtenida mediante recopilación directa de la empresa San Pedro San Pablo a través de encuestas, entrevistas, observaciones, análisis de contenido y estudio de casos metodología y procedimiento

3.5.1. Metodología de desarrollo XP (Extreme Programing)

La programación extrema es una metodología muy rápida y ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado.

A. Análisis

- ✓ Se utiliza historias de usuarios: las necesidades, escritas por los usuarios, con la ayuda de los diseños, que quieren ser satisfechas con el sistema
- ✓ Se crean los planes de entrega, los cuales estiman el tiempo de desarrollo de las historias de usuario.
- ✓ Se llevan a cabo la planificación de iteración: identificar las historias de usuarios que se van a desarrollar en una iteración especifica.

B. Diseño

- ✓ Se escoge una metáfora de sistema, esto para facilitar el manejo consistente de los nombres de las clases y los métodos.
- ✓ Se proponen soluciones a problemas técnicos o de diseño.
- ✓ Se ignoran las funcionalidades extra que podrían incorporarse al proyecto, centrar en lo principal.
- ✓ Se remueve la redundancia, se eliminan las funcionalidades no necesarias y se renuevan los diseños obsoletos.

C. Desarrollo

- ✓ Se utilizan estándares para escribir el código.
- ✓ Se crean las pruebas antes de empezar a codificar, lo cual hará más sencillas y efectivas las pruebas.
- ✓ Se realiza en equipos de trabajo y luego se llevó a cabo una integración paralela (debido a esta integración no se garantiza la consistencia y la calidad a necesidades de hacer pruebas exhaustivas)
- ✓ Se deja la optimización para el final. Una vez que el código requerido este completo.

D. Pruebas

- ✓ Se crean pruebas de aceptación a partir de las historias de usuario.
- ✓ El cliente es responsable de revisar, tanto las pruebas de aceptación, como los resultados obtenidos al ser estas aplicadas.
- ✓ Para el caso de la utilización del ISO – 9126 está desarrollado en el anexo N° 3 donde se llenó la ficha de evaluación y de esa forma se tomó la decisión que el sistema es factible.

Tabla 6

Ficha de Evaluación de la calidad del Producto Estándar ISO – 9126

INDICADORES	PUNTUACIÓN				
	1	2	3	4	5
1.FUNCIONALIDAD					
Adecuación: la capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas específicas y objetivos de los usuarios.					
Exactitud: la capacidad del producto de software para proporcionar los resultados o efectos correctos y con el grado de precisión acordado.					
Interoperabilidad: la capacidad del producto de software para interactuar con uno o más sistemas específicos.					
Seguridad: referido a la capacidad del producto software para proteger la información y los datos.					
Conformidad: la capacidad del producto software para adaptarse a los estándares, convenciones o regulaciones en leyes y prescripciones relativos a la funcionalidad.					
2.FIABILIDAD					
Madurez: la capacidad del producto software para evitar fallos provocados por errores en el software.					
Tolerancia o Fallos: la capacidad del producto software para mantener un nivel de rendimiento determinado en caso de defectos en el software o incumplimiento en su interfaz.					
Recuperabilidad: la capacidad del producto software para restablecer un determinado nivel de rendimiento y recuperar los datos afectados directamente en caso de ocurrir un fallo.					
Conformidad: la capacidad del producto software para adaptarse a los estándares, convenciones o regulaciones referidas a la fiabilidad.					
3.USUABILIDAD					
Comprensibilidad: la capacidad del producto software para permitir al usuario que entienda si el software es adecuado, y como debe utilizarse para determinadas tareas y bajo ciertas condiciones de uso.					
Facilidad de Aprendizaje: la capacidad del producto software para permitir al usuario aprender su aplicación.					
Atracción: la capacidad del producto software para atraer al usuario.					
Conformidad: la capacidad del producto software para adaptarse a estándares, convenciones, guías de estilo y regulaciones relacionadas con la usabilidad.					
Operabilidad: la capacidad del producto software para permitir que el usuario lo opere y lo controle.					
4.EFICACIA					
Comportamiento Temporal: la capacidad del producto software para proporcionar tiempos de respuesta y procesamiento apropiados cuando realiza sus funciones bajo condiciones determinadas.					
Utilización de Recursos: la capacidad del producto software para utilizar cantidades y tipos de recursos apropiados cuando el software realiza su función bajo determinadas condiciones.					
Conformidad: la capacidad del producto software para adaptarse a estándares o convenciones relacionadas con la eficacia.					
5.MANTENIBILIDAD					
Analizabilidad: capacidad del producto software de diagnosticar sus deficiencias o causas de fallos, o de identificar las partes que deben ser modificadas.					
Confiabilidad: capacidad del producto software de permitir implementar una modificación específica. La implementación incluye los cambios en el diseño, el código y la documentación.					
Estabilidad: capacidad del producto software de evitar los defectos inesperados de las modificaciones.					
Facilidad de Prueba: capacidad del producto software de permitir validad las partes modificadas.					
Conformidad: capacidad del producto software de cumplir los estándares o convenciones relativas a la mantenibilidad.					
6.PORTABILIDAD					
Adaptabilidad: la capacidad del producto software para ser adaptado para ambientes determinados sin realizar acciones o aplicar medios, más que los proporcionados para este propósito para el software considerado.					
Facilidad de Instalación: la capacidad del producto software para ser instalado en un ambiente determinado.					
Coexistencia: la capacidad del producto software para coexistir con otro software independientemente en un ambiente común compartiendo recursos.					
Reemplazabilidad: la capacidad del producto software para ser utilizado en lugar de otro producto de software para el mismo propósito en el mismo ambiente.					
Conformidad: la capacidad del producto software para adaptarse a estándares relacionados con la portabilidad.					
SUB TOTALES					
TOTAL					

FUENTE: ficha de evaluación ISO – 9126

Tabla 7

Escala Valorativa (Escala de Likert)

INDICADOR CUALITATIVO	VALOR
DEFICIENTE	1
MALO	2
REGULAR	3
BUENO	4
MUY BUENO	5

Tabla 8

Cuadro De Decisiones ISO 9126

CLASIFICACIÓN	INTERVALO	DECISIÓN
A) INACEPTABLE	[27 - 54 >	
B) MINIMAMENTE ACEPTABLE	[54 - 81 >	
C) ACEPTABLE	[81 - 95 >	
D) CUMPLE LOS REQUISITOS	[95 – 122 >	
E) EXCEDE LOS REQUISITOS	[122 – 135>	

3.6. Requerimientos del sistema

Especificación de requisitos de software SRP 830 IEEE

Fecha	Revisión	Autor	Verificado dep. Calidad.
15/01/2019	si	Renan Abelardo Palli Mamani	Verificado por el gerente dela empresa

Documento validado por las partes en fecha: 18/05/2019

Por la comunidad	Por la universidad
Centro poblado Iscayapi – Arapa	Universidad Nacional del Altiplano Puno

3.6.1. Requerimientos fundamentales

Se han definido para el sistema, los siguientes puntos más relevantes que el software debe poder realizar.

R1: Deben existir perfiles para el ingreso al sistema, este le dará mayor seguridad a la información que este maneje.

R2: El sistema debe estar en la capacidad de registrar nuevos productos.

R3: El sistema debe estar en la capacidad de poder editar datos de productos.

R4: Por necesidad debe existir la posibilidad de consultar datos de los productos.

R5: Modulo para generar las fichas de los nuevos productos y clientes.

R6: Modulo para generar recibos y facturas de pago.

R7: El sistema debe contar con modulo para generar reportes de los productos, clientes entre otros.

3.6.2. Requerimientos no Funcionales:

- ✓ Aplicación multiplataforma.
- ✓ Interfaz agradable para fácil entendimiento del software.
- ✓ Disponibilidad del sistema de encontrarse disponible todos los días.
- ✓ Portabilidad estará diseñado en un lenguaje multiplataforma.
- ✓ Mantenimiento y escalabilidad diseñado pensando en el crecimiento del sistema.

3.7. Requerimientos técnicos

Hardware

- ✓ Microprocesador INTEL 3.4Ghz.
- ✓ RAM de 4Gb.
- ✓ Tarjeta de video 128 Mb(16Mb)*
- ✓ Disco duro con espacio disponible de 10 MB (6Mb)*
- ✓ Monitor LED 18”
- ✓ Unidad de almacenamiento (CDROM y/o USB)

Software

- ✓ Plataforma Windows 7,8, 10 Profesional ®
- ✓ Servidor Web: App Server
- ✓ XAMPP
- ✓ CakePHP
- ✓ MySql – Workbench
- ✓ Dia
- ✓ Google Chrome
- ✓ Mozilla Firefox

CAPÍTULO IV
RESULTADOS Y DISCUSIÓN

Después de describir los resultados antes de implantar el sistema denominado *Desarrollo de un sistema de ventas para la planta de criadero de truchas Arapa S.A.C. 2018,* se discutió los resultados en función a los objetivos planteados.

4.1. Análisis del sistema

Para el desarrollo del sistema de ventas SISVEN, el primer paso fue analizar los datos e información de la planta criadero de truchas Arapa S.A.C. donde nos topamos que sus datos los guardaban en hojas de cálculo Excel. En varios de los casos la utilización de estas hojas de cálculo ha causado más demoras y pérdida de información y es así como se empezó con el diseño del sistema.

4.1.1. Análisis de requisitos del sistema

En esta parte se tomó en cuenta las historias de los usuarios que en este caso serían el administrador y todos los trabajadores quienes describieron de cómo les gustaría que sea el software que se va a construir. Para poder recolectar estas historias de usuario se han demoran 2 semanas.

Definición de roles

Dada la coyuntura de la investigación, y la disponibilidad de recursos humamos, el investigador ha asumido los roles de directa relación con el desarrollo del sistema. Solo se han tomado en consideración los roles más importantes según el desarrollo de la presente investigación.

- **Programador:** El investigador asume el rol de programador por tal motivo es el encargado de escribir el código fuente necesario para la implementación del sistema SISVEN.
- **Cliente:** El gerente de la empresa cumple el rol de cliente, define las especificaciones del sistema e influye en el desarrollo sin ejercer control, define las pruebas funcionales.
- **Tester:** Este rol es también asumido por el desarrollador con el fin de apoyar al cliente en la preparación y realización de pruebas también está encargado de explicar los resultados al equipo.
- **Tracker:** El investigador analiza la información sobre la marcha del proyecto sin afectar demasiado el proceso.
- **Entrenador:** El investigador, es el responsable global del proyecto también es el encargado de verificar que se estén aplicando correctamente las guías XP.

4.1.2. Funcionalidades, requisitos de tipo de usuario

En el sistema se tiene un solo administrador y/o usuario:

Administrador: Toda persona con una cuenta y accesos autorizados al sistema realiza funciones tales como: registro de nuevos usuarios, monitoreo del funcionamiento del sistema y notificación de los posibles errores a presentarse.

4.1.3. Casos de uso para el administrador y/o usuario

La propuesta de una interfaz está dirigida a desarrollar una herramienta tecnológica que facilite la integración de los miembros de la empresa Arapa S.A.C.

Utilizando las tecnologías que ofrece; para ello se realizó un modelado de las situaciones y convivencias que tienen los actores de la empresa Arapa S.A.C., a continuación se muestra un diagrama de casos de uso para mostrar el modelado de negocio.

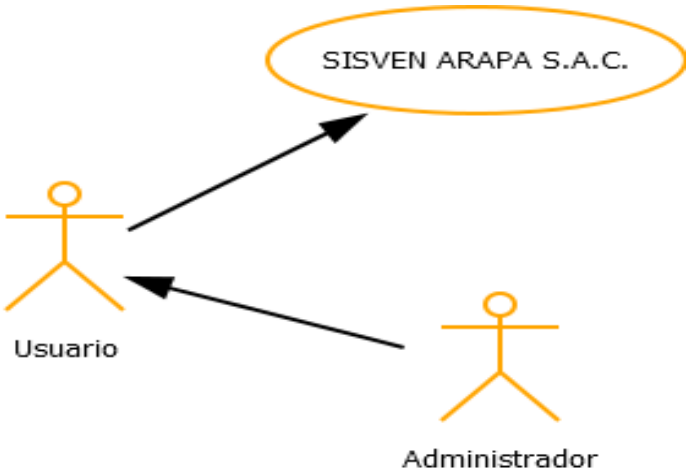


Figura 6. Diagrama de caso de uso para el Administrador y/o Usuarios

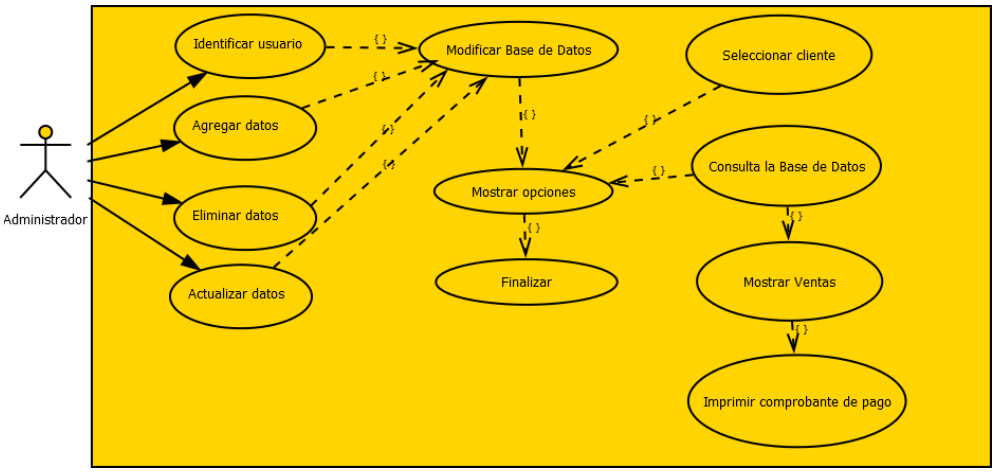


Figura 7. Diagrama de casos del sistema de ventas

4.2. Diseño

El propósito del diseño es de crear la solución propuesta que se planteó en la primera parte, se ha tomado en consideración el especial énfasis que hace XP en los diseños simples y claros. Para un mejor entendimiento de las propiedades del sistema y aplicando los diagramas UML.

4.2.1. Actor

Se indica un actor según el rol que desempeña al interactuar con el sistema de ventas.

- **Administrador** persona que pertenece a la empresa y además tiene atributos para modificar y actualizar la información del sistema.

4.2.2. Casos de uso

En este punto describiremos las actividades que realizaron los personajes o entidades que participaron en un diagrama de caso de uso los cuales se denominaron actores.

4.2.3. Diagrama de casos de uso

El propósito del diseño es de crear la solución propuesta. La primera parte comprende el diseño en alto nivel de la arquitectura justificando la elección de un patrón arquitectónico. Respecto a la interfaz gráfica, se mencionan los patrones y estándares adoptados para uniformizar el aspecto visual y la interacción a partir de los requerimientos del sistema los cuales se diseñaran a continuación.

4.2.3.1. Diagrama de casos de uso actores del sistema

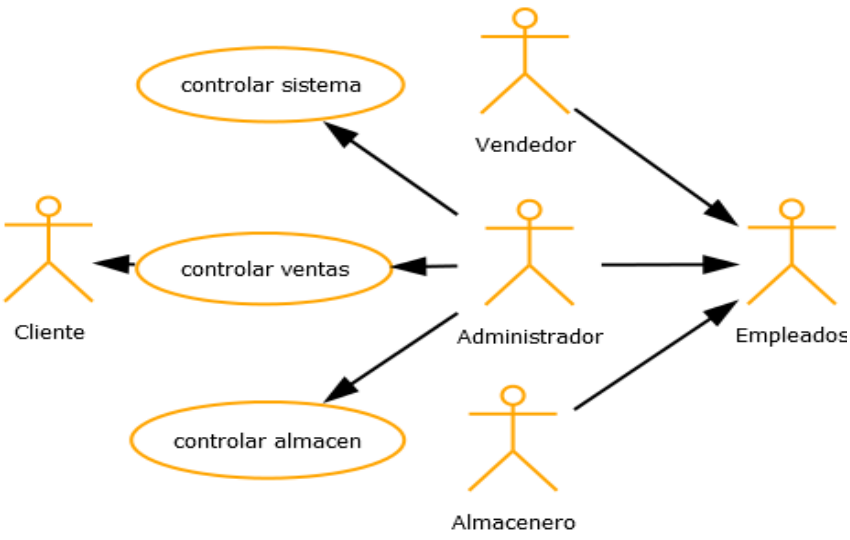


Figura 8. Modelo de casos de uso

4.2.3.1.1. Diagrama de casos de uso actores del sistema

Las especificaciones de los casos de usos del negocio tratan lo siguiente:

4.2.3.1.2. Gestionar el sistema

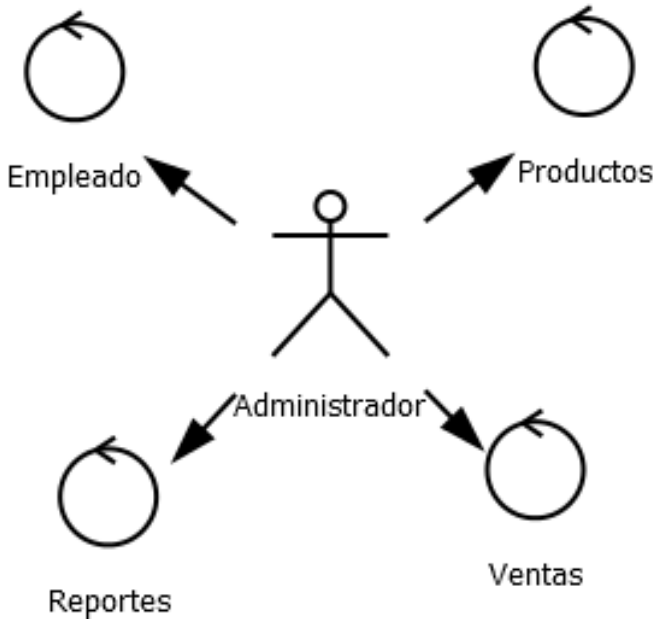


Figura 9. Modelo de Objetos Gestionar Sistema

- Descripción:

  Se tendrá en cuenta que se contara con un responsable el cual velara por el rendimiento efectivo del sistema. Asimismo, quien brindara soporte a las acciones administrativas propias del sistema.

- Objetivo:

  Almacenar información base de los clientes

- Responsable:

  Administrador

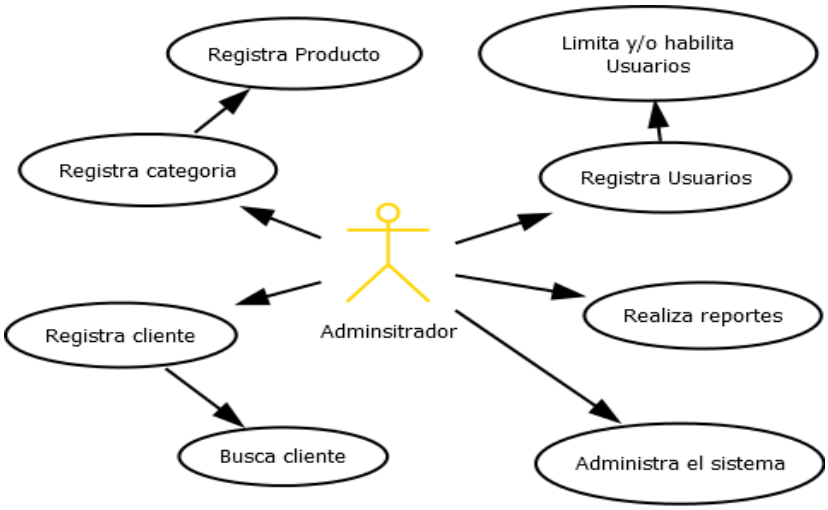


Figura 10. Modelo de Casos de Uso de Requerimiento Gestionar Sistema

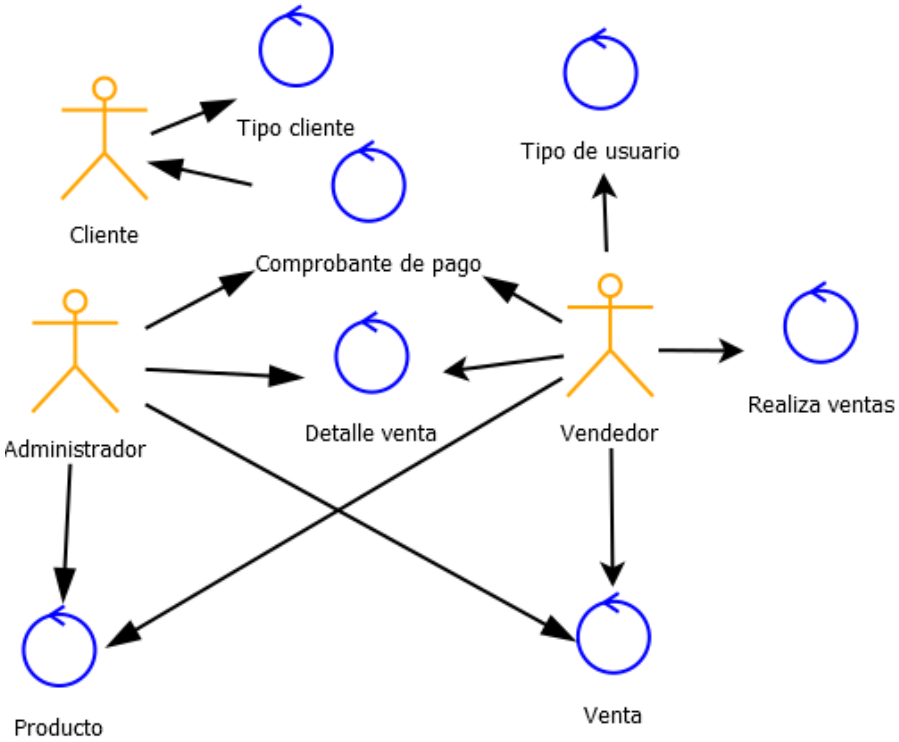


Figura 11. Modelo de Objetos Controlar Ventas

- Descripción:

  Es el caso de uso del negoción que permite ingresar y modificar los clientes, es donde brinda el soporte al proceso de ventas en sí que realizan diariamente la entidad. Puesto que con este componente se registran las ventas y reportes de las ventas realizadas.

- Objetivos:

  Almacenar información de ventas.

- Responsable

  Administrador, vendedor y cliente.

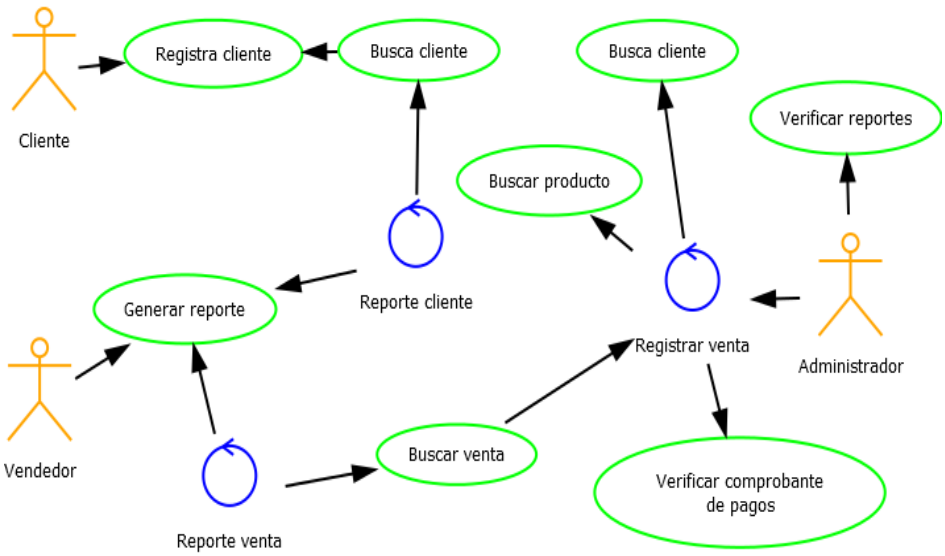


Figura 12. Modelo de Caso de Uso de Requerimiento Controlar Venta

4.2.3.1.3. Controlar almacén

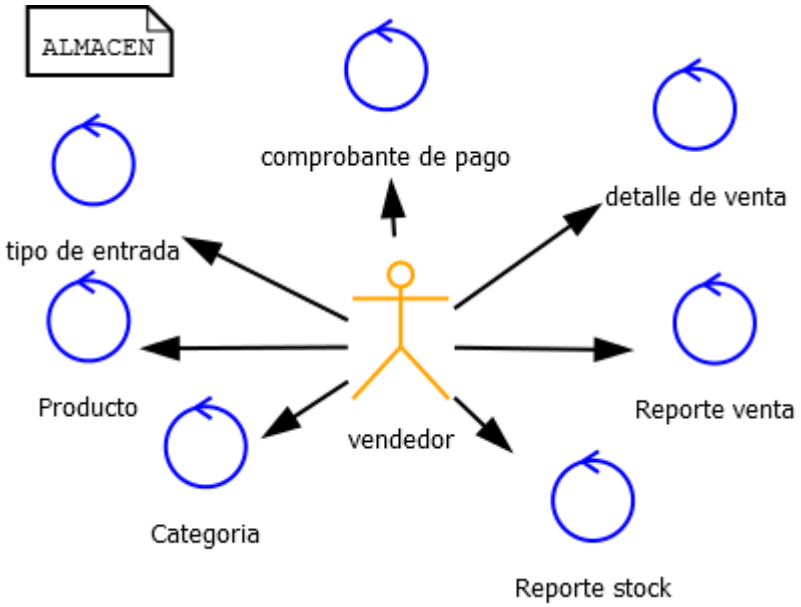


Figura 13. Modelo de Objetivo Controlar Almacén

- Descripción:

  Es el caso de uso del negocio que permite controlar de flujo de entrada y salida de productos de almacén. Así mismo permite controlar el stock de los mismos para así realizar requerimientos de pedidos a proveedores.

- Objetivos:

  Controlar el stock de los productos.

- Responsable

  Almacenero.

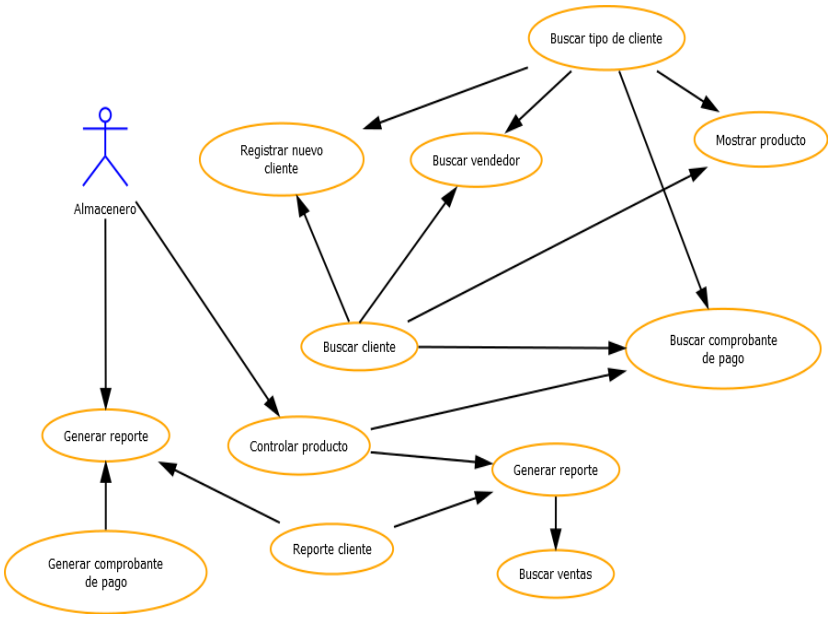


Figura 14. Modelo de Casos de Uso de Requerimiento Controlar Almacén

4.2.3.2. Diagrama de clases

Todos los casos de uso anteriores se representan en diagramas de clases, comenzando por el modelado del administrador y/o usuario que representan a las entidades de negocio identificadas en la etapa de análisis con sus atributos y tipos de datos utilizados la cual se muestra en la Figura 15.

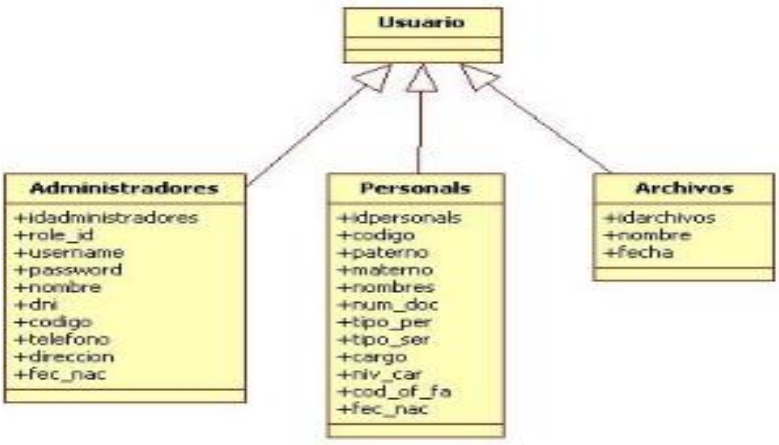


Figura 15. Diagrama de clases para el Administrador y/o Usuario

4.2.3.2.1. Diagrama de clase para el Administrador y/o usuario

Empezamos por la entidad Administrador y/o Usuario el cual está asociada a los casos de uso: LOGIN

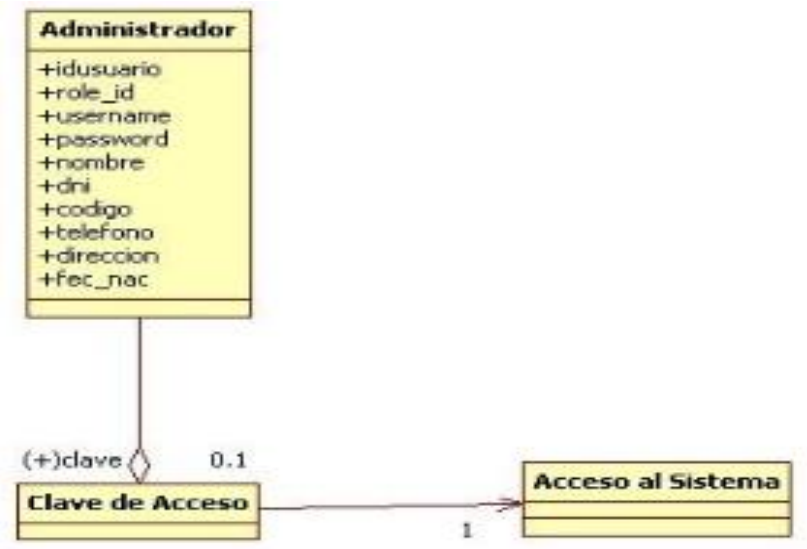


Figura 16. Diagrama de clase Login el Administrador y/o Usuario

Según se observa en la Figura, el administrador y/o usuario a través de una clave de acceso puede ingresar al sistema, esta clave de acceso tiene dos únicas posibilidades (cierto = 1 o falso = 0) de lo cual se define el acceso a la interfaz la cual solamente se hace por una sola entidad con esta descripción.

De la misma forma también el administrador puede registrar más usuarios para que puedan acceder al sistema, también puede modificar información y así mismo borrar usuarios.

4.2.3.3. Diagrama de actividades

La siguiente etapa en el modelo de análisis basado en UML comprende el modelado de las acciones a ejecutarse en el sistema.

4.2.3.3.1. Diagramas de Actividades para el Administrador y/o Usuario

Comenzando con el actor Administrador que es el encargado de actualizar lo concerniente a la información se puede resumir las actividades a desarrollar en tres actividades generales que comprenden: Realizar informe de Registros, Actualizaciones, Actividades de Mantenimientos y Reportes. Esto se presenta en la siguiente figura.

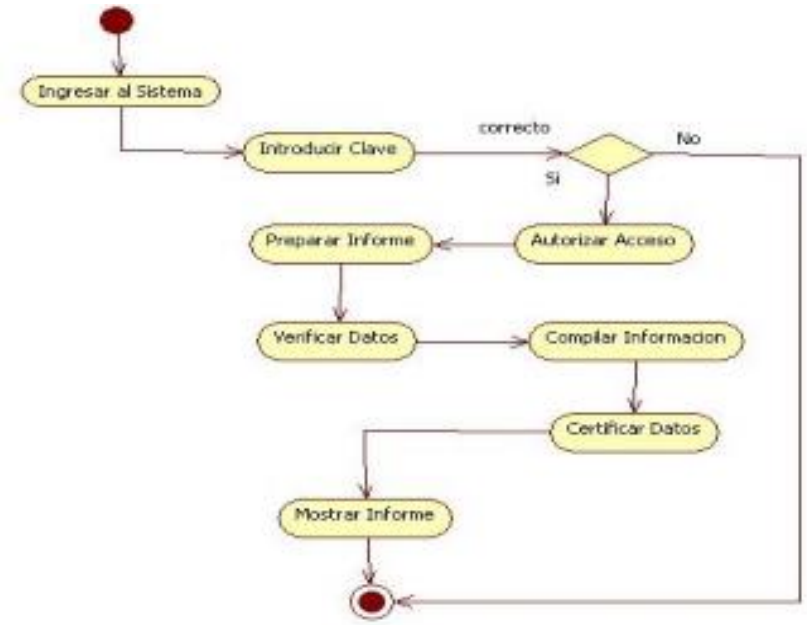


Figura 17. Diagrama de actividades para el Administrador y/o Usuario

Por su parte, el Actor Administrador y/o Usuario cumple sus funciones dentro del Sistema en este proceso se muestran en el diagrama siguiente:

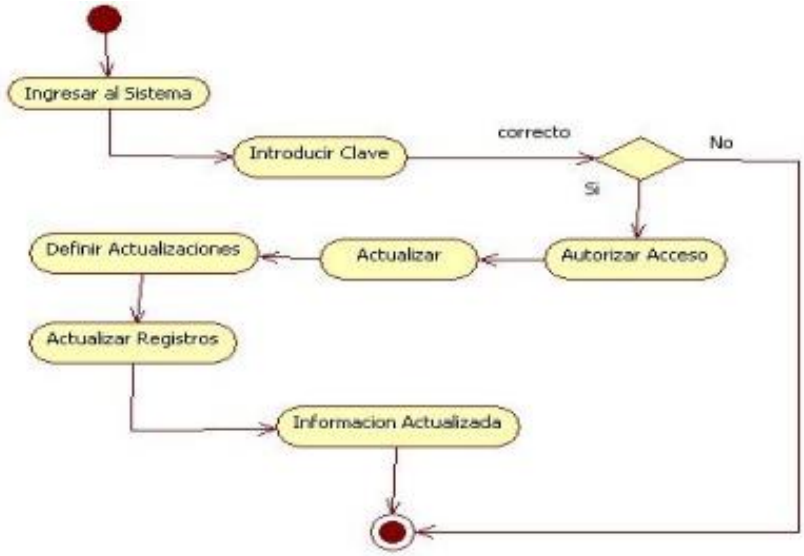


Figura 18. Diagrama de actividades, Actualizar Registros

4.2.3.4. Diagrama de integraciones administrador

En el diagrama de interacciones se representa la forma como los actores (Administrador) y los (Objetos) se comunican entre sí según los eventos del sistema. Para desarrollar el siguiente diagrama de integración se utiliza la información obtenida de los Casos de Uso:

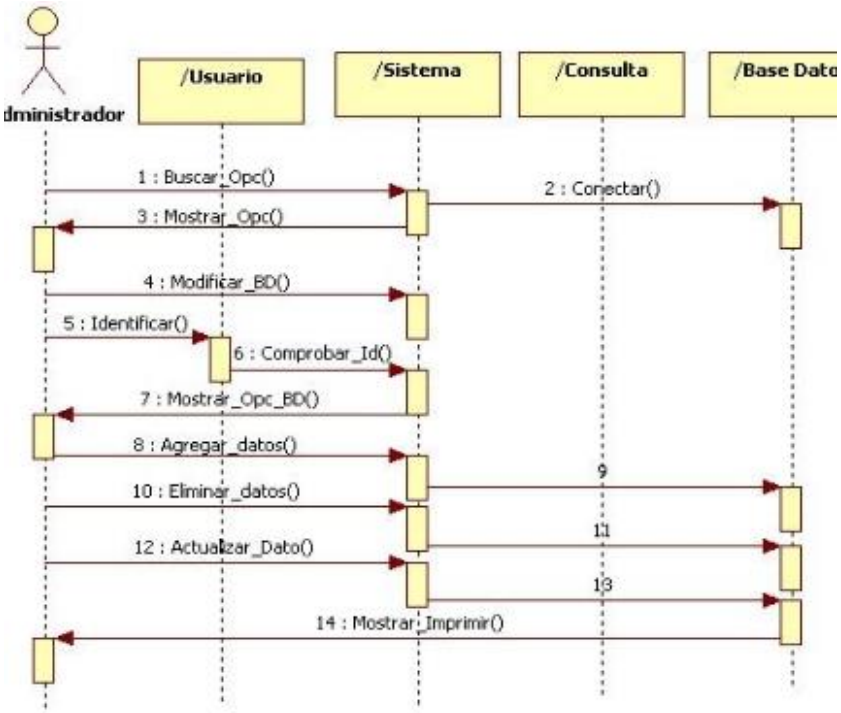


Figura 19. Diagrama de interacciones (Administrador)

4.2.3.5. Diagrama de componentes

Diagrama de componentes de sistemas de información que se muestra en la siguiente figura:

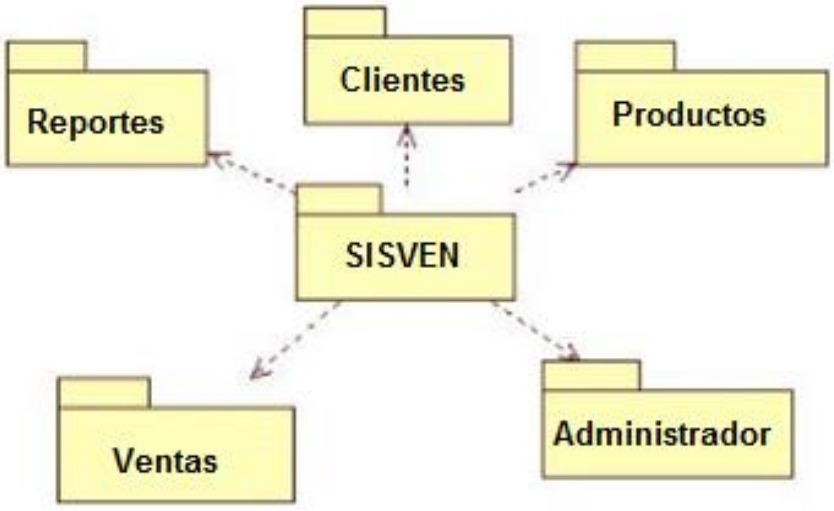


Figura 20. Componentes del Sistema de Ventas SISVEN

4.2.3.5.1. Sistema

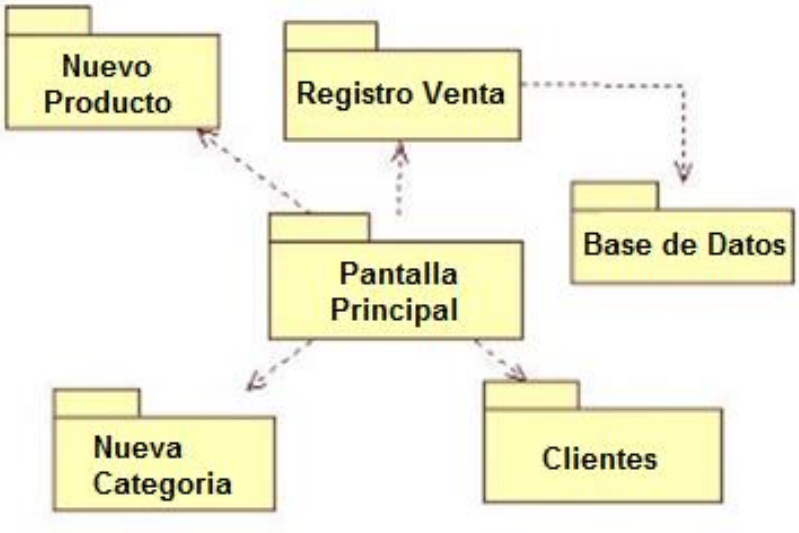


Figura 21. Módulos del Sistema

4.2.3.6. Arquitectura de datos del sistema

Se utiliza la arquitectura clásica para sistemas de información, la cual consiste en tres niveles y son los siguientes:

- **NIVEL 1:** Presentación (interfaz), donde se encuentra las pantallas y menús.
- **NIVEL 2:** Lógica de la aplicación, se refiere a las tareas y reglas de los procesos.
- **NIVEL 3:** Almacenamiento, se encuentra la base de datos del sistema.

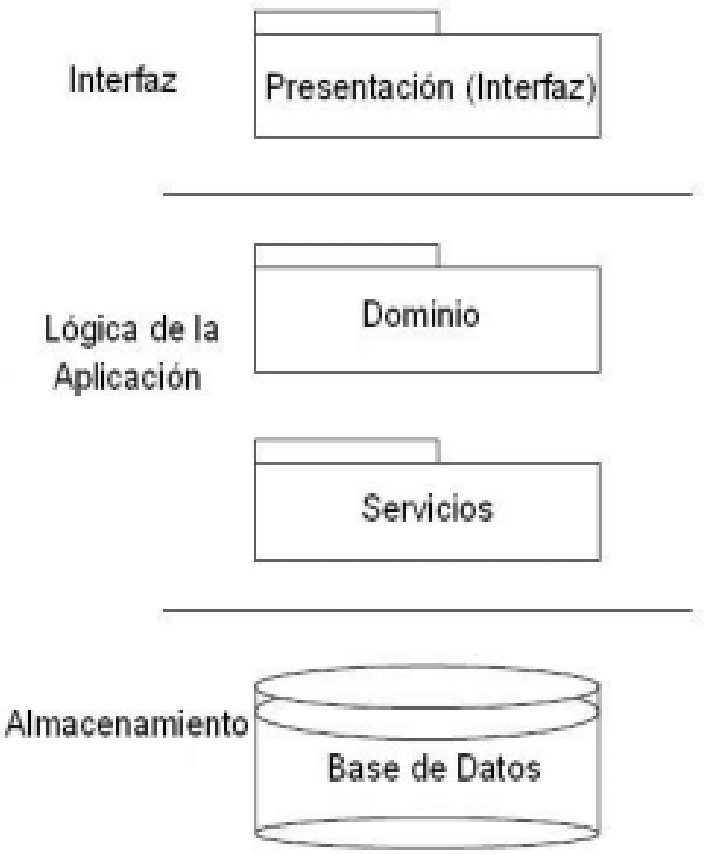


Figura 22. Arquitectura del sistema

4.2.3.6.1. Modelamiento y funciones del sistema

Diagrama de casos de uso, que describe el tipo de dato como entrada al proceso de análisis del software. Un caso de uso es, en esencia, una interacción típica entre un usuario y un sistema de cómputo; y emplearemos el termino actor para llamar al cliente, cuando desempeña ese papel con respecto al sistema.

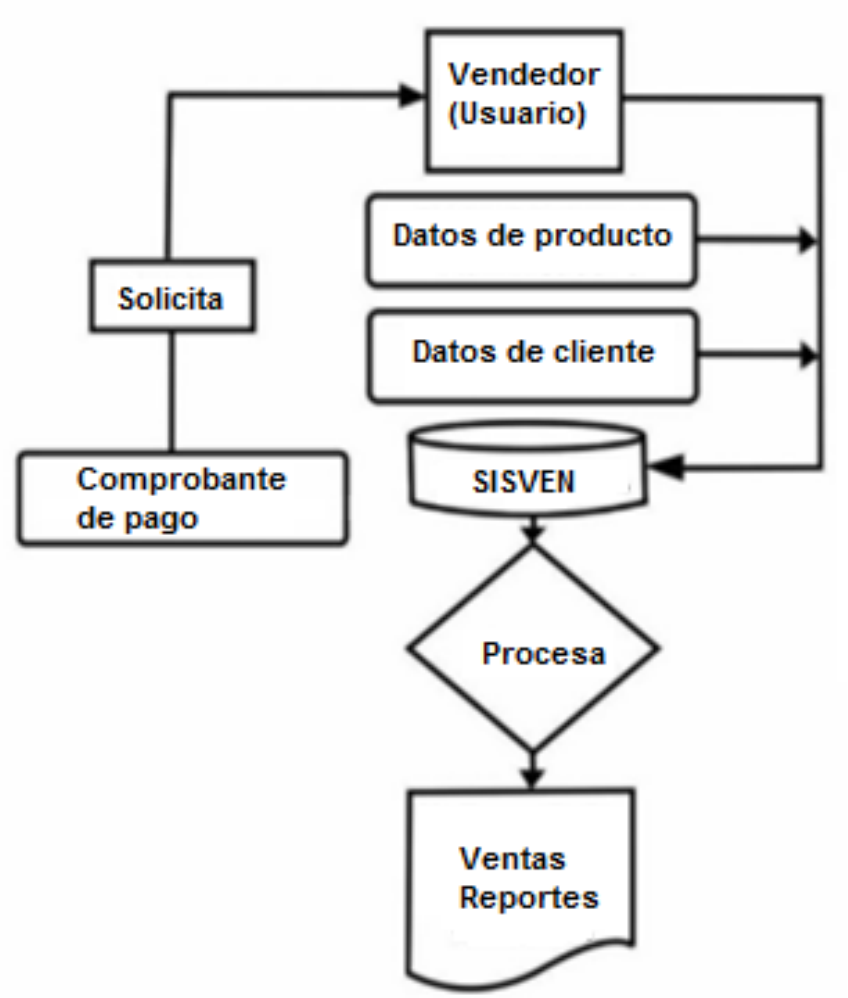


Figura 23. Modelamiento y funciones del Sistema SISVEN

4.2.3.7. Tablas de archivos

Seguidamente se procede a establecer la arquitectura de datos del sistema, la cual está compuesta de tres (03) tablas.

4.2.3.7.1. Tabla para el administrador y/o usuario

Tabla 9
Administrador y/o Usuarios

#	Columna	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	id	int(11)			No	*Ninguna*
2	nombres	varchar(100)	latin1_swedish_ci		Sí	*NULL*
3	apellidos	varchar(100)	latin1_swedish_ci		Sí	*NULL*
4	telefono	varchar(20)	latin1_swedish_ci		Sí	*NULL*
5	email	varchar(100)	latin1_swedish_ci		Sí	*NULL*
6	username	varchar(45)	latin1_swedish_ci		Sí	*NULL*
7	password	varchar(100)	latin1_swedish_ci		Sí	*NULL*
8	rol_id	int(11)			Sí	*NULL*
9	estado	tinyint(1)			Sí	*NULL*

Fuente: tabla base de datos xampp

4.2.3.7.2. Tabla para clientes

Tabla 10
Clientes

#	Columna	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	id	int(11)			No	*Ninguna*
2	nombre	varchar(100)	latin1_swedish_ci		Sí	*NULL*
3	telefono	varchar(20)	latin1_swedish_ci		Sí	*NULL*
4	direccion	varchar(100)	latin1_swedish_ci		Sí	*NULL*
5	tipo_cliente_id	int(11)			Sí	*NULL*
6	tipo_documento_id	int(11)			Sí	*NULL*
7	num_documento	varchar(45)	latin1_swedish_ci		Sí	*NULL*
8	estado	tinyint(1)			Sí	*NULL*

Fuente: tabla base de datos xampp

4.2.3.7.3. Tabla ventas

Tabla 11
Tabla 11Ventas

#	Columna	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	id	int(11)			No	*Ninguna*
2	fecha	date			Sí	*NULL*
3	subtotal	varchar(45)	latin1_swedish_ci		Sí	*NULL*
4	igv	varchar(45)	latin1_swedish_ci		Sí	*NULL*
5	descuento	varchar(45)	latin1_swedish_ci		Sí	*NULL*
6	total	varchar(45)	latin1_swedish_ci		Sí	*NULL*
7	tipo_comprobante_id	int(11)			Sí	*NULL*
8	cliente_id	int(11)			Sí	*NULL*
9	usuario_id	int(11)			Sí	*NULL*
10	num_documento	varchar(45)	latin1_swedish_ci		Sí	*NULL*
11	serie	varchar(45)	latin1_swedish_ci		Sí	*NULL*

Fuente: tabla base de datos xampp

4.2.3.8. Modelos entidad relación

En este apartado, se analizara el diagrama entidad / relación, los datos se han modelado según el diagrama E/R mostrando en la figura.

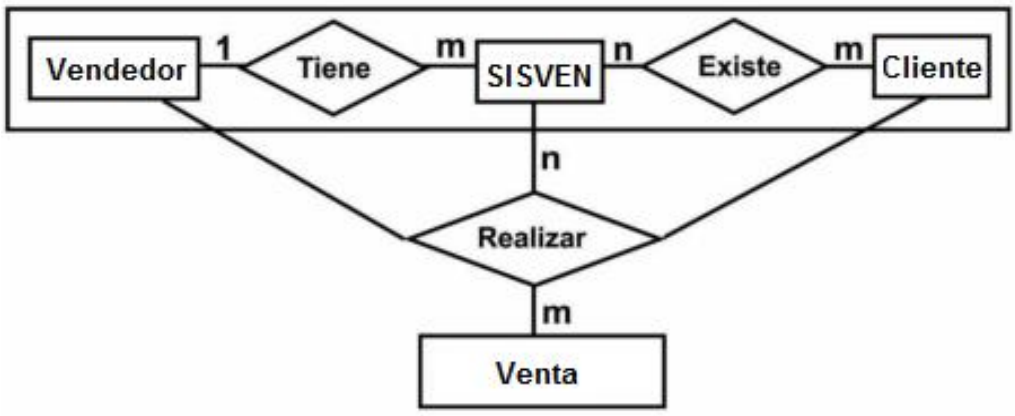


Figura 24. Modelo Entidad Relación (E - R)

En la empresa de criadero de truchas San Pedro San Pablo hay un tiempo de venta hacia un cliente el grafico anterior nos ayuda a entender mejor la funcionalidad del sistema

4.2.4. Implementación del prototipo de sistema de administración de contenidos

4.2.4.1. Organización de la Aplicación:

Antes de desarrollar la aplicación veremos la organización de la aplicación en cuanto a carpetas.

Para desarrollar la implementación del sistema, descargamos el framework CodeIgniter, el cual nos permite organizar la aplicación según el contexto **MVC** (Modelo Vista Controlador). Las carpetas que contiene son:

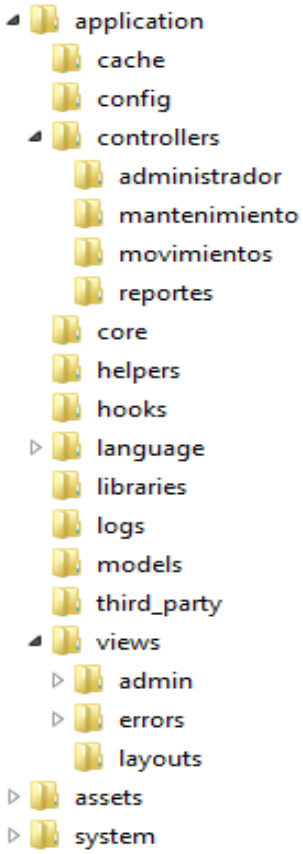


Figura 25. Organización de la aplicación según CodeIgniter

En la carpeta appllication se tienen las subcarpetas controllers, models y views, las cuales son las más importantes y donde se encuentran los archivos de desarrollo del sistema.

La carpeta instalar, sólo la utilizaremos al realizar la instalación del sistema, para mayor seguridad se debe borrar después de tal instalación.

La carpeta systems es el corazón del framawork donde se encuentran sus archivos principales

4.3. Desarrollo

En el presente capítulo describiremos la metodología para la implementación del sistema de ventas (SISVEN), patrones de implementación y la arquitectura utilizada, también se tiene los prototipos de las pantallas a implementar.

Durante el desarrollo se ha generado una serie de ficheros de código, documentación que han precisado de cierto tiempo para completarse.

4.3.1. Generación y proceso de formularios en CakePHP

El poder de CakePHP reside en su simplicidad y velocidad. Una de las aplicaciones más comunes que se utilizan en este lenguaje son los formularios de CakePHP, por su parte CakePHP poses conocimientos generales de PHP y conocimientos básicos de programación orientada a objetos para el desarrollo de los formularios

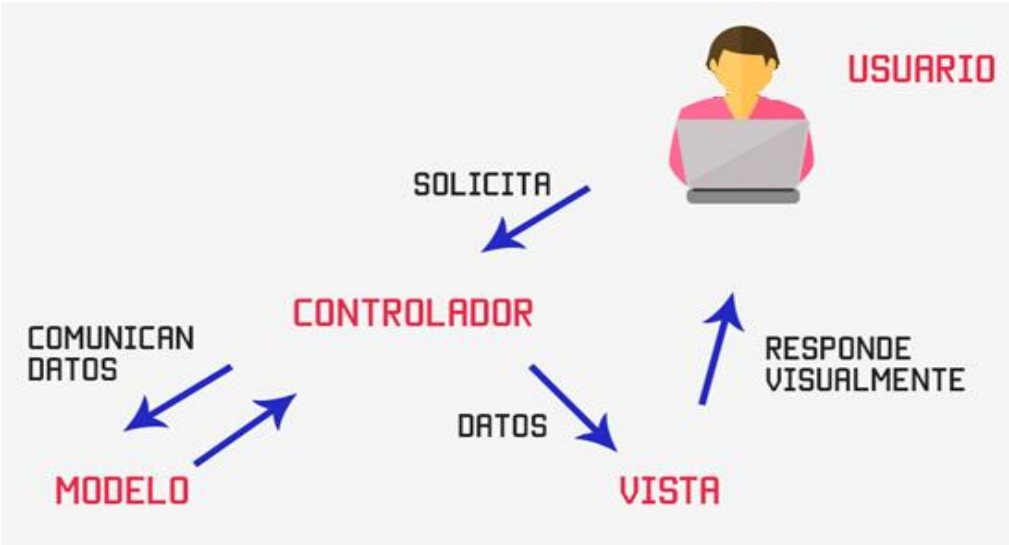


Figura 26. Arquitectura de Software (CakePHP)

- **Arquitectura de software**:

  Se utilizó el MVC, para separar componentes de nuestra aplicación dependiendo de la responsabilidad que tienen. Es decir, una parte del código no debe de saber qué es lo que hace toda la aplicación, sólo debe de tener una responsabilidad.

  En web, el MVC funciona cuando el usuario manda una petición al navegador, digamos que se requiere visualizar los reportes de venta del mes, 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 los reportes de venta del mes, le pide al modelo la información del reporte.

  El modelo, se encarga de los datos de la app, consulta la base de datos, obtiene todos los reportes de venta del mes, la información del reporte de venta, el modelo responde al controlador con los datos que pidió, porque el controlador pide datos, y el modelo responde con los datos solicitados).

  Una vez el controlador tiene los datos de reportes de venta del mes, se los manda a la vista, la vista aplica los estilos, organiza la información y construyó la página SISVEN.

- **Codificación**

  La utilización de CakePHP, nos obliga a adoptar cierto tipo de estándares en su codificación para este caso, se hacen uso de los estándares de codificación, de tal forma que así se reduce la complejidad de los trabajos de mantenimiento y escalado del sistema cumpliéndose así lo recomendado por XP.

  La arquitectura MVC se adapta bien al sistema de SISVEN para responder estas peticiones al modelo, vistas y el controlador.

```php
<?php
class Clientes extends CI_Controller {
    private $permisos;
    public function __construct(){
        parent::__construct();
        $this->permisos = $this->backend_lib->control();
        $this->load->model("Clientes_model");
    }

    public function index()
    {
        $data  = array(
            'permisos' => $this->permisos,
            'clientes' => $this->Clientes_model->getClientes(),
        );
        $this->load->view("layouts/header");
        $this->load->view("layouts/aside");
        $this->load->view("admin/clientes/list",$data);
        $this->load->view("layouts/footer");

    }
    public function add(){

        $data = array(
            "tipoclientes" => $this->Clientes_model->getTipoClientes(),
            "tipodocumentos" => $this->Clientes_model->getTipoDocumentos()
        );

        $this->load->view("layouts/header");
        $this->load->view("layouts/aside");
        $this->load->view("admin/clientes/add",$data);
        $this->load->view("layouts/footer");
    }
```

Figura 27. Estándar de codificación para controladores

```php
<?php
class Clientes_model extends CI_Model {
    public function getClientes(){
        $this->db->select("c.*,tc.nombre as tipocliente, td.nombre as tipodocumento");
        $this->db->from("clientes c");
        $this->db->join("tipo_cliente tc", "c.tipo_cliente_id = tc.id");
        $this->db->join("tipo_documento td", "c.tipo_documento_id = td.id");
        $this->db->where("c.estado","1");
        $resultados = $this->db->get();
        return $resultados->result();
    }

    public function getCliente($id){
        $this->db->where("id",$id);
        $resultado = $this->db->get("clientes");
        return $resultado->row();

    }
    public function save($data){
        return $this->db->insert("clientes",$data);
    }
    public function update($id,$data){
        $this->db->where("id",$id);
        return $this->db->update("clientes",$data);
    }

    public function getTipoClientes(){
        $resultados = $this->db->get("tipo_cliente");
        return $resultados->result();
    }

    public function getTipoDocumentos(){
        $resultados = $this->db->get("tipo_documento");
        return $resultados->result();
    }
}
```

Figura 28. Estándar de codificación para modelos

```php
<?php
class Permisos_model extends CI_Model {

    public function getPermisos(){
      $this->db->select("p.*,m.nombre as menu,r.nombre as rol");
      $this->db->from("permisos p");
      $this->db->join("roles r","p.rol_id = r.id");
      $this->db->join("menus m","p.menu_id = m.id");
      $resultado=$this->db->get();
      return $resultado->result();
    }
    public function getMenus(){
      $resultado = $this->db->get("menus");
      return $resultado->result();
    }

    public function save($data){
      return $this->db->insert("permisos",$data);
    }
    public function getPermiso($id){
      $this->db->where("id",$id);
      $resultado = $this->db->get("permisos");
      return $resultado->row();
    }
    public function update($id,$data){
      $this->db->where("id",$id);
      return $this->db->update("permisos",$data);
    }

    public function delete($id){
      $this->db->where("id",$id);
      return $this->db->delete("permisos");
    }

}
```

Figura 29. Estándar de codificación para Permisos

Para la codificación del sistema, se hace uso del patrón MVC que nativamente proporciona CakePHP.

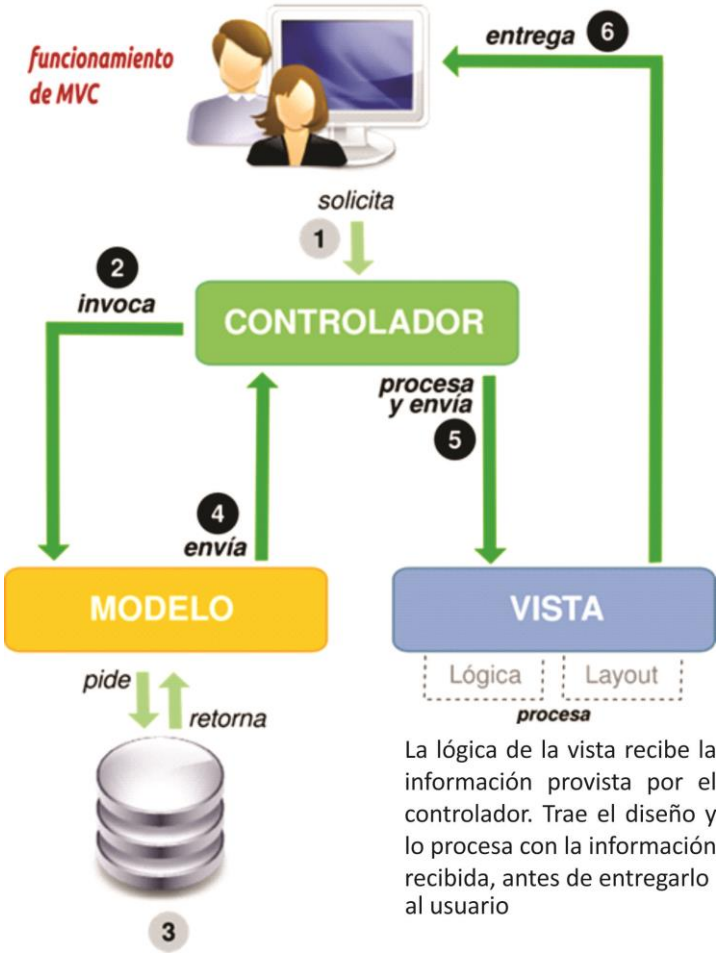


Figura 30. Funcionamiento del patrón MVC

En el trabajo que se realizó se detalla lo siguiente:

- **Modelo:** Aquí fue donde se recolectaron los datos e información para poder desarrollar el sistema SISVEN, Fue donde se realizó las encuestas entrevistas y demás para así realizar la implementación del sistema SISVEN en la planta de criaderos de truchas Arapa S.A.C.
- **Vista:** Aquí desarrollo la interfaz del sistema SISVEN, simple y amigable para los usuarios.
- **Controlador:** En esta etapa se hizo las correcciones necesarias en el modelo y vista, de esta manera se hizo un software de calidad y buen funcionamiento para los trabajadores de mencionada empresa.

4.3.1.1. Pantalla Principal o índex

Al acceder al dominio reservado por la institución en este caso el servidor local (http://localhost/truchas/) se inicia una pantalla. En ella se muestra el título se va a mantener constante durante toda la navegación que haga el usuario.

Figura 31. Pantalla principal o Índex

4.3.1.2. Ventana de acceso

Con esta pantalla se inicia la ejecución del SISTEMA VENTAS (SISVEN), para lo cual el administrador encargado deberá ingresar el nombre de usuario y la clave de acceso, para acceder al de inscripción. Cabe recalcar que si no se conoce la clave administrador, negara el acceso al sistema.

Figura 32. Ventana de acceso

4.3.1.3. Ventana de nuevos usuarios

La figura, muestra la ventana de inscripción de registro de nuevos usuarios. Esta ventana se ocupa de almacenar la información de nuevos usuarios; y además nos muestra los elementos que lo componen.

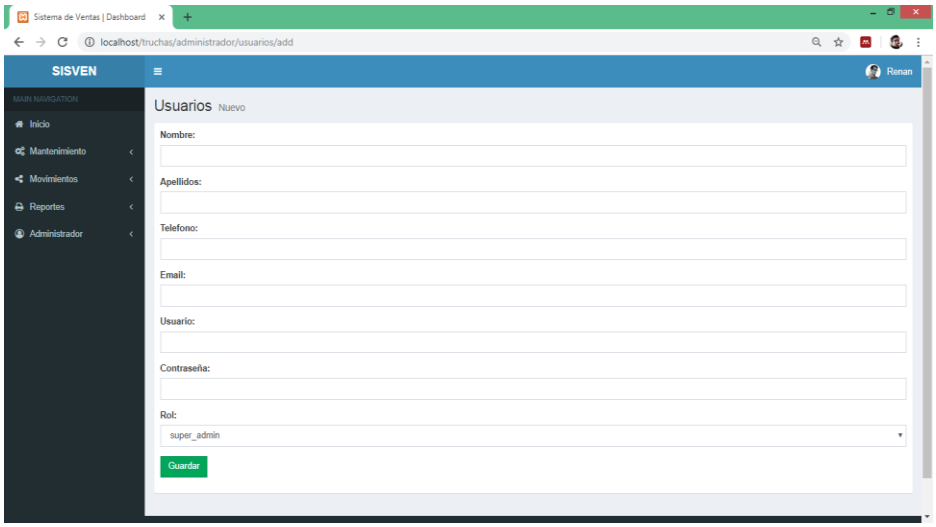


Figura 33. Ventana de Nuevos Usuarios

4.3.1.4. Ventana de nuevos clientes

La figura, muestra la ventana de Ingreso, el registro de nuevos clientes, para que de esta manera puedan acceder al sistema

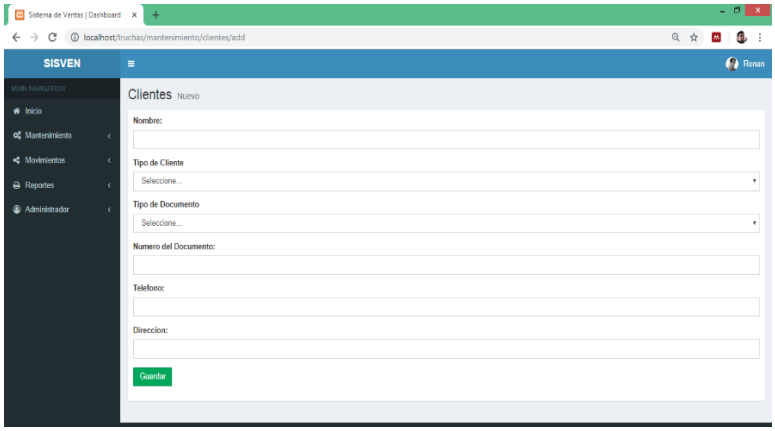


Figura 34. Ventana de Nuevos Clientes

4.3.1.5. Ventana de nuevos productos

En la figura se muestra la opción para poder agregar nuevos productos a la base de datos.

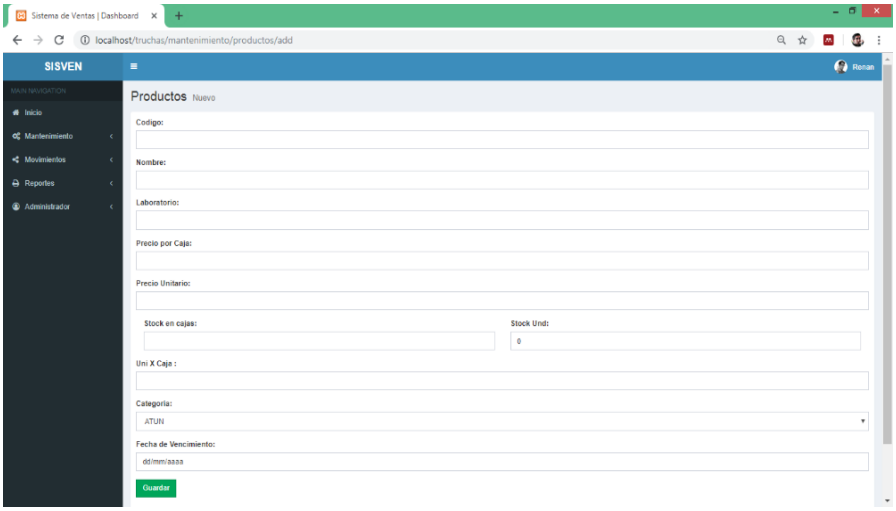


Figura 35. Ventana de Nuevos Productos

4.3.1.6. Ventana de ventas

En la figura muestra las ventas realizadas hasta la fecha

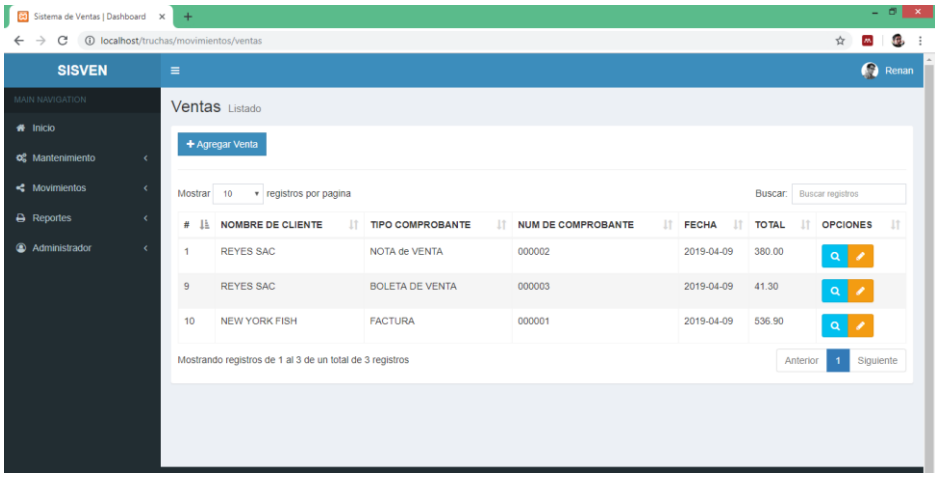


Figura 36. Ventana de Ventas

- **Reportes del sistema**

En el grafico mostramos el comprobante de pago que genera el sistema SISVEN.

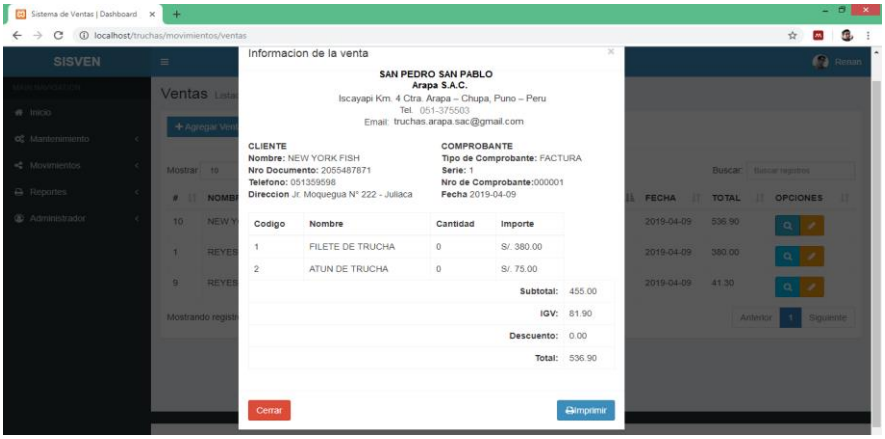


Figura 37. Reportes del Sistema

En el grafico muestra el total de ventas realizadas

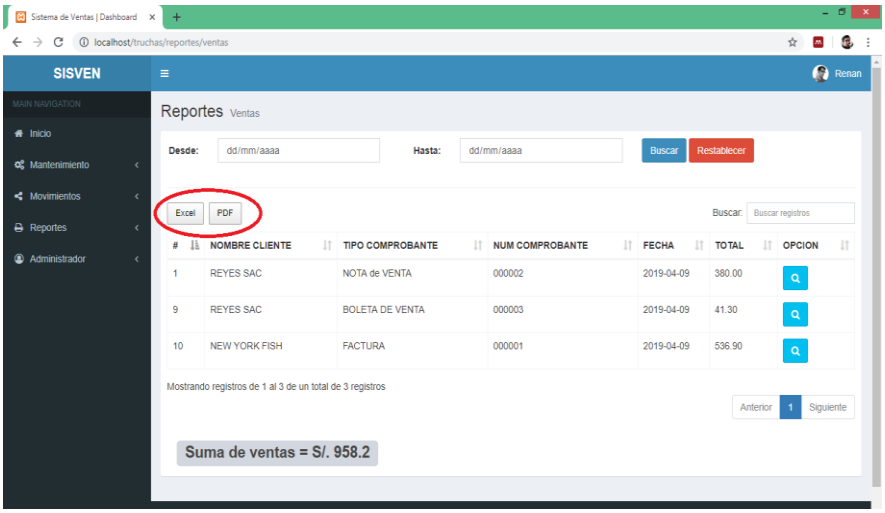


Figura 38. Reportes de las Ventas.

4.3.1.7. Base de datos

4.3.1.7.1. Construcción de la base de datos

Es el componente principal de un sistema de información, para la construcción de la base de datos se utilizó el programa MySQL, el cual está construida por 12 tablas; las mismas que muestran en la siguiente figura:

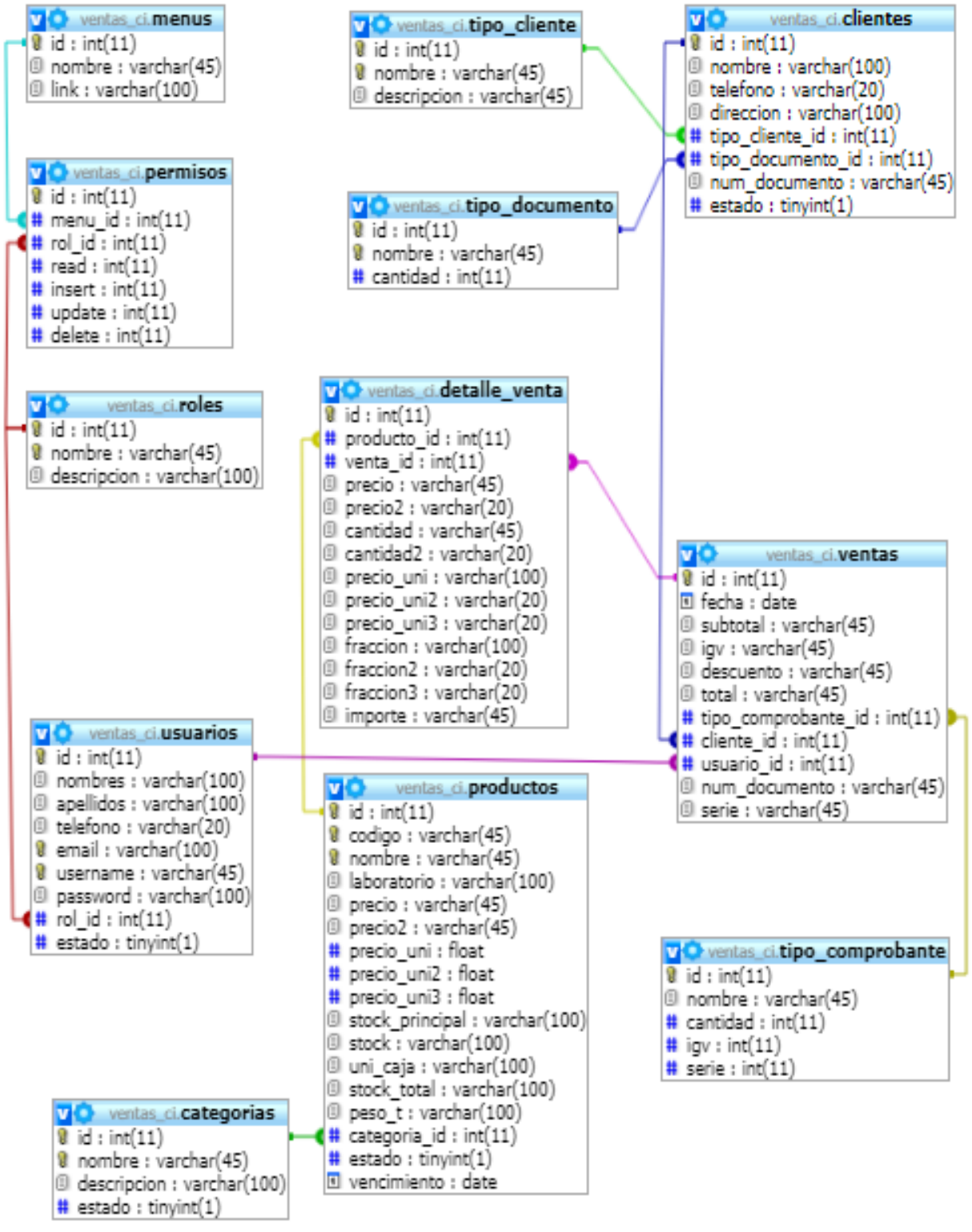


Figura 39. Base de Datos SISVEN

4.3.1.8. Implementación

a. **CakePHP:** Se utilizó CakePHP para el marco de desarrollo (frameword) rápido para PHP, libre, de código totalmente abierto, se trata de una estructura que sirve de base a los programadores para que se puedan crear aplicaciones. El objetivo primordial es trabajar de forma estructurada y rápida, sin pérdida de flexibilidad.

b. **PHP:** Como se mencionó, PHP permite generar páginas Web con contenido dinámico, por lo cual es necesario recordar cómo funcionan estas páginas.

   Las páginas Web son documentos que se pueden visualizar a través de Internet, que pueden incluir texto, imágenes, ligas de hipertexto y otros medios.

c. **PHPMyAdmin**: Es un programa de libre distribución en PHP, creado por una comunidad sin ánimo de lucro, que sólo trabaja en el proyecto por amor al arte. Es una herramienta muy completa que permite acceder a todas las funciones típicas de la base de datos MySQL a través de una interfaz Web muy intuitiva. La aplicación en si no es más que un conjunto de archivos en PHP que podemos copiar en un directorio de nuestro servidor Web.

4.4. Prueba

A. Pruebas no convencionales

Son pruebas que consisten en las revisiones técnicas formales que se realizaron en las etapas de análisis y diseño del sistema, que corrigen errores básicamente de:

- Omisiones y ambigüedad en las definiciones de clases y jerarquías, así como en las relaciones.
- Inconsistencias en la elaboración de Diagrama de Casos de Uso, Interacción, Clases y Actividades.

B. Pruebas convencionales

Son pruebas que se pueden ejecutarse o probarse, los cuales se realizan en la etapa de la implementación del sistema como son las pruebas de caja negra y caja blanca.

Prueba de caja blanca: Esta prueba de software, fue desarrollada durante la construcción de cada módulo. Mediante esta prueba se garantiza que el prototipo del sistema de control, cumplió con:

- Ejecutar todos los caminos independientes de cada módulo.
- La estructura de los datos es compatible.
- Ninguno de los bucles es infinito o su ejecución es por demás.
- Cuenta con todas las decisiones lógicas necesarias.
- Dar mayor prioridad a las consultas de búsqueda

Prueba de caja negra: Esta prueba de software se aplicó en el desarrollo de los módulos así como también estuvieron terminados y enlazados entre ellos para su funcionamiento como sistema. Mediante esta prueba aseguramos que el Prototipo de Sistema SISVEN, no tiene errores de:

- Procedimientos o funciones incorrectas.
- Los pie de reporte muestran información de continuación
- Errores de entrada y salida.
- Errores de rendimiento.
- Errores de inicialización y finalización.
- Los resultados de la consulta muestra lo requerido

C. Prueba del sistema

Esta fase de prueba del sistema se realiza mediante el método Prueba Basado en Escenarios, con el fin de descubrir errores de interfaz y errores del procesamiento de datos al nivel de los resultados esperados. La prueba se concentra en lo que el usuario hace, interacción del usuario con el sistema. La validación de la funcionalidad integra del sistema se prueba con los datos de la población de trabajadores de la empresa, esto permite verificar la certidumbre de los resultados proporcionados.

Tabla 12

Prueba del Sistema el Antes y Después

Antes de implementar el sistema	**Después de implementar el sistema**
Emisión de reportes mensuales	Emisiones de reportes diarias, semanales, mensuales.
Guardar información de clientes en libros físicos o plantillas de Excel.	La información se guarda en la base de datos y puede ser impresa el momento que se requiera
Se produce constantes errores en el llenado de los comprobantes de pago.	No se produce errores al momento de generar los comprobantes de pago.
No hay seguridad	Tiene seguridad
Solo el personal autorizado tiene acceso a la información de las ventas	El Super_Admin y el administrador tienen acceso a todo la información los usuarios registrados están restringidos a lo que corresponda.
La información de la empresa es vulnerable a cualquier tipo de percances.	Los trabajadores de la empresa son los únicos conocedores de su clave de acceso al sistema
Si hay un nuevo producto se tiene que usar un cuaderno nuevo o un Excel nuevo	El sistema tiene para que se pueda ingresar mediante el mismo sistema los nuevos productos y así se guardara automáticamente en la base de datos.

Las investigaciones de Oliver Gomez (2017) y Elvis Aliaga (2017) concluyen que la metodología XP se adapta al tema de investigación que realizaron. En mi investigación también se utilizó la misma metodología porque es la más usada en el desarrollo de sistemas de información.

Por lo tanto concluimos que el sistema de ventas implementado en la planta de criaderos de truchas Arapa S.A.C. mejoró notablemente para los encargados de ventas y almacén en beneficio al crecimiento de la empresa.

4.5. Prueba de hipótesis para diferencia de dos medias muéstrales

Realizando prueba de hipótesis para dos medias muéstrales y poder así sustentar uno de los objetivos específicos planteado: Analizar el tiempo en la eficiencia en las ventas y atención al cliente antes y después de la implementación del sistema

4.5.1. Planteamiento de Hipótesis

Ho: *Si la probabilidad obtenida* ***P-valor*** *<=* α*; se rechace Ho (Se Acepta* H_1*) (Hipótesis nula)*

H_1***:*** *Si la probabilidad obtenida* ***P-valor*** *>* α*; no rechace Ho (Se Acepta* H_0*) (Hipótesis alterna)*

4.5.2. Fijar el Nivel de Significancia (α)

$\alpha = 0.05$ *al 95 % de nivel de confianza)*

4.5.3. Prueba Estadística

Seguidamente haremos la prueba de t-student para dos muestras relacionadas, Los tiempos recolectados están en minutos.

Tabla 13

Encuesta del Antes y Después del Sistema.

TIEMPO	ANTES	DESPUES
1	8.00	1.00
2	10.00	1.00
3	15.00	1.50
4	12.00	1.50
5	6.00	1.00
6	17.00	3.00
7	13.00	2.00
8	7.00	1.00
9	12.00	2.00
10	9.00	1.00

- Se hizo el cálculo de la prueba t-student utilizando el paquete estadístico SPSS.

Tabla 14

Estadísticas de muestras relacionadas

		Media	N	Desviación estándar	Media de error estándar
Par 1	Datos antes de la implementación del sistema	**10.9000**	10	3.54181	1.12002
	Datos después de la implementación del sistema	**1.5000**	10	.66667	.21082

Tabla 15

Prueba de muestras relacionadas

		Diferencias emparejadas					t	gl	Sig. bilateral
		Media	Desviación estándar	Media de error estándar	95% de intervalo de confianza de la diferencia Inferior	Superior			
Par 1	Datos antes de la implementación del sistema - Datos después de la implementación del sistema	9.40000	2.99815	.94810	7.25525	11.54475	9.915	9	**.000**

4.5.4. Decisión

$$\textbf{P-valor} = 0.000 \quad < \quad \alpha = 0.05$$

Hay una diferencia significativa en las medias de los tiempos del antes y después de la implementación del sistema. Por lo cual se concluye que la implementación del sistema SI redujo el tiempo de venta. De hecho los tiempos de ventas en promedio bajaron de 10.90 a 1.50 minutos

CONCLUSIONES

- Se ha logrado desarrollar satisfactoriamente un sistema de ventas para la planta de criadero de truchas Arapa S.A.C. del distrito de Arapa Provincia Azángaro, Región puno.

- Se ha logrado analizar las ventas y/o Stock con el sistema de ventas SISVEN con lo siguiente: toma decisiones al momento de que el sistema tenga poco stock en un producto (Algoritmos avanzados), permite ingresar nuevas categorías y nuevos productos, el sistema funcionara con total normalidad; realiza reportes diarios, semanales, mensuales, anuales al requerimiento del personal; también tiene una interfaz amigable y fácil de manipular.

- Al analizar el tiempo de demora del antes y después de haber implementado el sistema concluimos que hay una diferencia significativa en las medias de los tiempos del antes y después de la implementación del sistema.

$$\textbf{P-valor} = 0.000 \quad < \quad \alpha = 0.05$$

Por lo cual se concluye que la implementación del sistema SI redujo el tiempo de venta. De hecho los tiempos de ventas en promedio bajaron de 10.90 a 1.50 minutos

- Se diseñó e implemento el sistema SISVEN utilizando la metodología Programación Extrema – XP porque es la más usada para este tipo de investigaciones y es la que más se adecua a mi trabajo.

RECOMENDACIONES

- En primer lugar se recomienda que si la empresa crezca y tenga más sucursales sería bueno implementar el sistema con todas las sucursales que tenga así el administrador pueda controlar todas las ventas desde cualquier parte.

- Al momento de mejorar el sistema sería excelente que se implementen sensores dentro de las jaulas de truchas para así poder controlar de manera exacta el stock de las truchas vivas y de esta manera poder llevar un bien control dentro y fuera del lago.

- Realizar la recolección de datos con otro tipo de instrumento de recojo de información para así poder comprobar cuál es mejor.

- Con los avances tecnológicos sería bueno implementar el sistema con otra metodología de implementación de sistemas de información. Para la presente investigación se tomó como herramienta de desarrollo software libre, sin embargo existen otros lenguajes de programación con licencia que se pueden utilizar para futuros proyectos y así incluir nuevas mejoras que fortalezcan su robustez y velocidad.

BIBLIOGRAFÍA

Abud Figueroa, M. A. (2012). *Calidad en la Industria del Software . La Norma ISO-25000*. 2–4.

Acosta, L. A. (2005). *Guía práctica para la sistematización de proyectos y programas de cooperación técnica Oficina Regional de la FAO para América Latina y el Caribe*. 29. Retrieved from http://www.fao.org/3/a-ah474s.pdf

Aliaga Payehuanca, E. A. (2016). *Universidad Nacional Del Altiplano Tesis*. 113. Retrieved from http://repositorio.unap.edu.pe/bitstream/handle/UNAP/9408/Rosa_Enriquez_Yuca.pdf?sequence=1&isAllowed=y

Booch, G., Rumbaugh, J., & Jacobson, I. (1999). El Lenguaje Unificado de Modelado. *Elements*, 30. https://doi.org/1852 - 4516

Carrera Jimenez, D. S. (2011). Análisis y diseño de un sistema de trámite de documentos de pago a proveedores vía intranet. *Pontificia Universidad Católica Del Perú*. Retrieved from http://tesis.pucp.edu.pe/repositorio/handle/123456789/343

Carrizo, D., & Rojas, J. (2018). Metodologías, técnicas y herramientas en ingeniería de requisitos: un mapeo sistemático Methodologies, techniques and tools in requirements engineering: a systematic mapping. *Revista Chilena de Ingeniería*, *26*(3), 473–485.

Cobo, Angel , Gomez, Patricia, P. D. y R. R. (2005). *PHP y MySQL Tecnologías para el desarrollo de aplicaciones web*. *1*, 497.

EcuRed. (2017). Lenguaje de programación (informática) - EcuRed. Retrieved April 8, 2019, from https://www.ecured.cu/Lenguaje_de_programación_(informática)#Lenguaje_de_Programaci.C3.B3n

Ginesta, M. G., & Peña González, Á. (2005). Software Libre. Ingeniería del software en entornos de SL. *Training*, 314.

Gomez Cutipa, O. (2017). *Universidad Nacional Del Altiplano Tesis*. 113. Retrieved from http://repositorio.unap.edu.pe/bitstream/handle/UNAP/9408/Rosa_Enriquez_Yuca.pdf?sequence=1&isAllowed=y

Kendall Kenneth, & Kendall Julie. (2005). *Análisis y Diseño de Sisyemas*. Retrieved from https://luiscastellanos.files.wordpress.com/2014/02/analisis-y-disenio-de-sistemas-kendall-kendall.pdf

Laura Murillo, R. P. (2014). *Universidad Nacional Del Altiplano - Puno Facultad Universidad Nacional Del Altiplano - Puno*. 1–22.

Meléndez Valladarez, S. M., Gaitan, M. E., & Pérez Reyes, N. N. (2016). *Sistema web de evaluación al desempeño docente Unan-Managua, empleando la metodologia ágil programación extrema, en el II semestre de 2015.* 146. Retrieved from http://repositorio.unan.edu.ni/1365/1/62161.pdf

Peña, A. (2006). *Ingeniería de Software : Una Guía para Crear Sistemas de Información* (A. P. A. D.R. © 2006, Ed.). Mexico.

Samaniego Pallaroso, M. A. (2012). *Artículo Científico - Desarrollo de un sistema vía Web para control de producción en la granja avícola Marco Antonio Vivanco Álvarez, aplicando la metodología FDD con herramientas libre*. Retrieved from http://repositorio.espe.edu.ec/xmlui/handle/21000/5298

Stallman, R. M. (2004). Software libre para una sociedad libre. In *Mapas* (Vol. 9). Retrieved from http://biblioweb.sindominio.net/pensamiento/softlibre/softlibre.pdf

Ticahuanca, C. E. S. (2017). *"Sistema De Gestión Comercial Aplicando Erp Para Grupo Perusis S.A.C."* 1–160. https://doi.org/10.1007/8904_2014_350

ANEXOS

Anexo 1. Encuesta antes de la implementación del sistema

ENCUESTA ANTES DE LA IMPLEMENTACIÓN DEL SISTEMA DE VENTAS (SISVEN)

Instrucciones: lea detenidamente cada uno de los ítems. Encierre en un círculo y/o marque con una (x) en cada ítem, solo una de las alternativas que mejor sugiera su opinión.

1. **¿Cuánto tiempo demora al realizar una venta?**

   a) 5 minutos a 10 minutos

   b) 10 minutos a 15 minutos

   c) 15 minutos a más

2. **¿Ha tenido alguna vez confusión de datos al generar el comprobante de pago?**

   a) Siempre

   b) De vez en cuando

   c) Nunca

3. **Cree Ud. Que la confusión de información perjudico el stock en almacén según su parecer es:**

   a) La tecnología

   b) El personal

   c) El tiempo

Anexo 2. Encuesta después de la implementación del sistema

ENCUESTA DESPUES DE LA IMPLEMENTACIÓN DEL SISTEMA DE VENTAS (SISVEN)

Instrucciones: lea detenidamente cada uno de los ítems. Encierre en un círculo y/o marque con una (x) en cada ítem, solo una de las alternativas que mejor sugiera su opinión.

1. **¿Cuánto tiempo demora en realizar una venta?**

   a) 1 minutos a 2 minutos

   b) 2 minutos a 3 minutos

   c) 3 minutos a más

2. **¿Ha tenido alguna vez confusión de datos al generar el comprobante de pago?**

   a) Siempre

   b) De vez en cuando

   c) Nunca

2. **Cree Ud. Que la confusión de información perjudico el stock en almacén según su parecer es:**

   a) Tecnología

   b) Personal

   c) Tiempo

Anexo 3. Ficha de evaluación ISO - 9126

Ficha de evaluación de la Calidad del producto Estándar ISO – 9126

INDICADORES	PUNTUACION				
	1	2	3	4	5
1. FUNCIONALIDAD					
Adecuación: la capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas específicas y objetivos de los usuarios.					X
Exactitud: la capacidad del producto software para proporcionar los resultados o efectos correctos y con el grado de precisión acordado.					X
Interoperabilidad: la capacidad del producto software para interactuar con uno o más sistemas específicos.				X	
Seguridad: referido a la capacidad del producto software para proteger la información y los datos			X		
Conformidad: la capacidad del producto software para adaptarse a los estándares, convenciones o regulaciones en leyes y prescripciones relativos a la funcionalidad				X	
2. FIABILIDAD					
Madurez: la capacidad del producto software para evitar fallos provocados por errores en el software.				X	
Tolerancia a Fallos: la capacidad del producto software para mantener un nivel de rendimiento determinado en caso de defectos en el software o incumplimiento de su interfaz.					X
Recuperabilidad: la capacidad del producto software para restablecer un determinado nivel de rendimiento y recuperar los datos afectados directamente en caso de ocurrir un fallo.					X
Conformidad: la capacidad del producto software para adaptarse a los estándares, convenciones o regulaciones referidas a la fiabilidad.				X	
3. USUABILIDAD					
Comprensibilidad: la capacidad del producto software para permitir al usuario que entienda si el software es adecuado, y como debe utilizarse para determinadas tareas y bajo ciertas condiciones de uso.				X	
Facilidad de Aprendizaje: la capacidad del producto software para permitir al usuario aprender su aplicación.				X	
Atracción: la capacidad del software para atraer al usuario.			X		
Conformidad: la capacidad del producto software para adaptarse a estándares, convenciones, guías de estilo y regulaciones relacionadas con la usabilidad.				X	
Operabilidad: la capacidad del producto software para permitir que el usuario lo opere y lo controle				X	
4. EFICIENCIA					
Comportamiento Temporal: la capacidad del producto software para proporcionar tiempos de respuesta y de procesamiento apropiados cuando realiza sus funciones bajo condiciones determinadas.				X	
Utilización de Recursos: la capacidad del producto software para utilizar cantidades y tipos de recursos apropiados cuando el software realiza su función bajo determinadas condiciones					X
Conformidad: la capacidad del producto software para adaptarse a estándares o convenciones relacionadas con la eficiencia.				X	
5. MANTENIBILIDAD					
Analizabilidad: capacidad del producto software de diagnosticar sus deficiencias o causas de fallos, o de identificar las partes que deben ser modificadas.					X
Confiabilidad: capacidad del producto software de permitir implementar una modificación especifica. La implementación incluye los cambios en el diseño, el código y la documentacion					X
Estabilidad: capacidad del producto software de evitar los defectos inesperados de las modificaciones.					X
Facilidad de Prueba: capacidad del producto software de permitir validar las partes modificadas.					X
Conformidad: capacidad del producto software de cumplir los estándares o convenciones relativas a la mantenibilidad.				X	
6. PORTABILIDAD					
Adaptabilidad: la capacidad del producto software para ser adaptado para ambientes determinados sin realizar acciones o aplicar medios, más que los proporcionados para este propósito para el software considerado			X		
Facilidad de Instalación: la capacidad del producto software para ser instalado en un ambiente determinado.				X	
Coexistencia: la capacidad del producto software para coexistir con otro software independiente en un ambiente común compartiendo recursos.				X	
Reemplazabilidad: la capacidad del producto software para ser utilizado en lugar de otro producto de software para el mismo propósito en el mismo ambiente.				X	
Conformidad: la capacidad del producto software para adaptarse a estándares relacionados con la portabilidad.					X
SUB TOTALES			9	56	50
TOTAL					

Indicadores:

INDICADOR CUALITATIVO	VALOR
Deficiente	1
Malo	2
Regular	3
Bueno	4
Muy Bueno	5

Cuadro de decisión ISO 9126

INDICADOR CUALITATIVO	VALOR	DECISIÓN
A) Inaceptable	[27-54>	
B) Mínimamente aceptable	[54-81>	
C) Aceptable	[81-95>	
D) Cumple los requisitos	[95-122>	115
E) Excede los requisitos	[122-135>	

Conclusión:

De acuerdo a los resultados de la validación de software se concluye que el sistema de ventas de truchas con 115 puntos según la escala de calificación del ISO - 9126

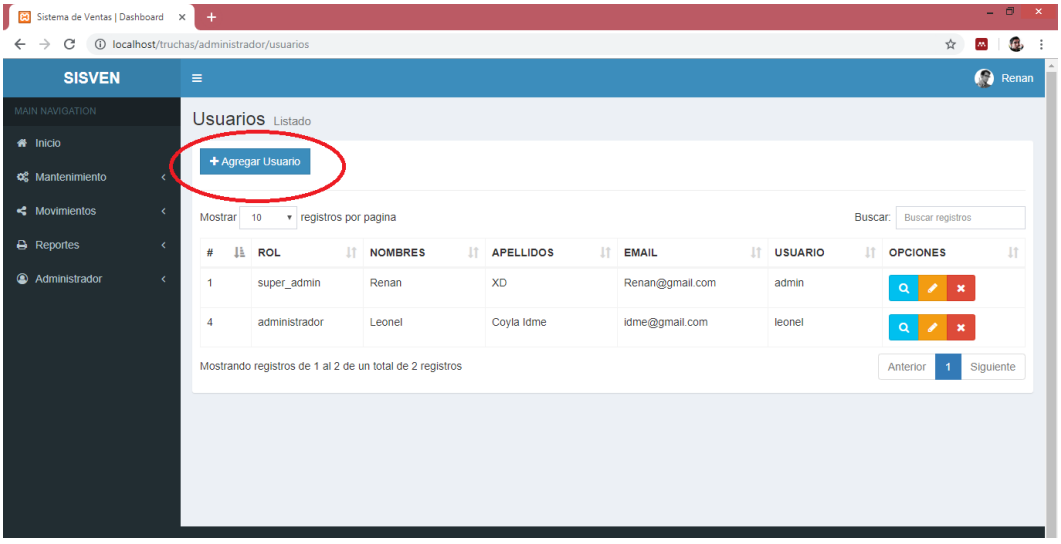
Anexo 4. Manual de usuario SISVEN

MANUAL DE USUARIO DEL SITEMA DE VENTAS (SISVEN)

El sistema de ventas (SISVEN) tiene una interfaz amigable para los usuarios.

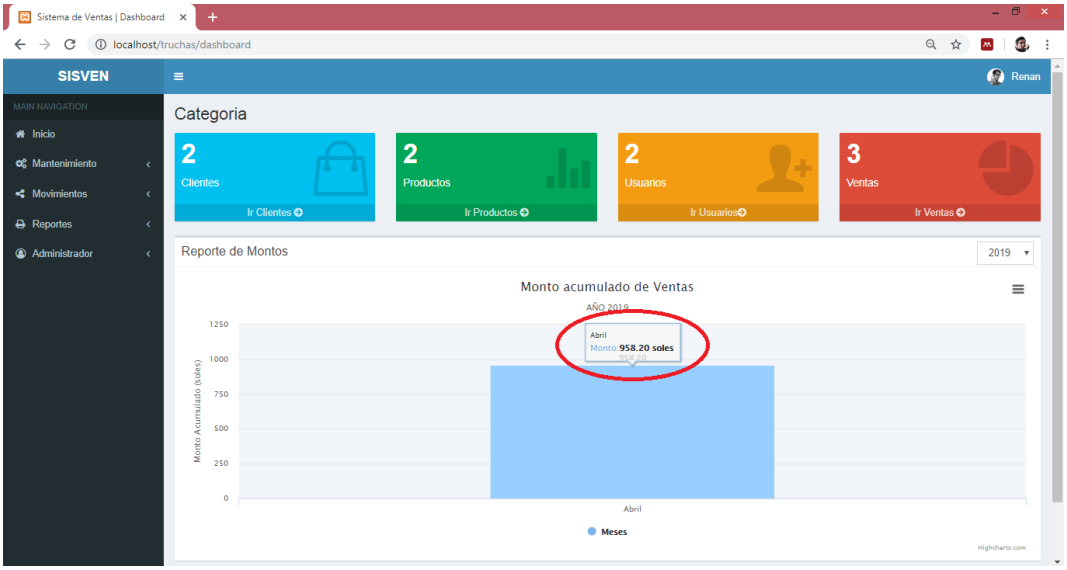
1) Lo primero que se tiene que hacer es acceder al sistema.

2) Solo en Super Admin y Admin pueden agregar nuevos usuarios para que puedan hacer uso del sistema

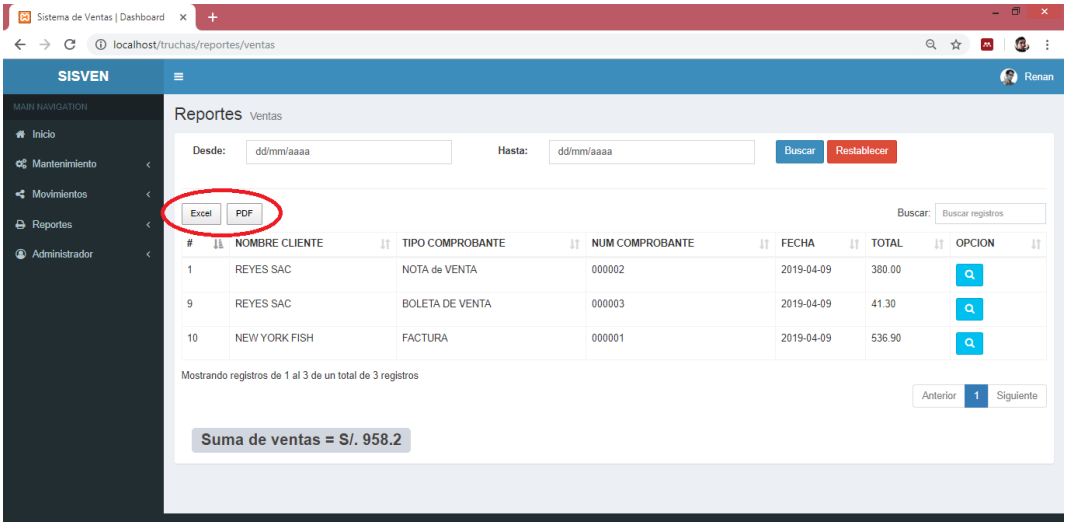


3) En la pantalla principal podemos observar bastantes opciones de colores para una mejor adaptación en poco tiempo.

La parte remarca nos muestra con un gráfico estadístico las ventas realizadas hasta la fecha

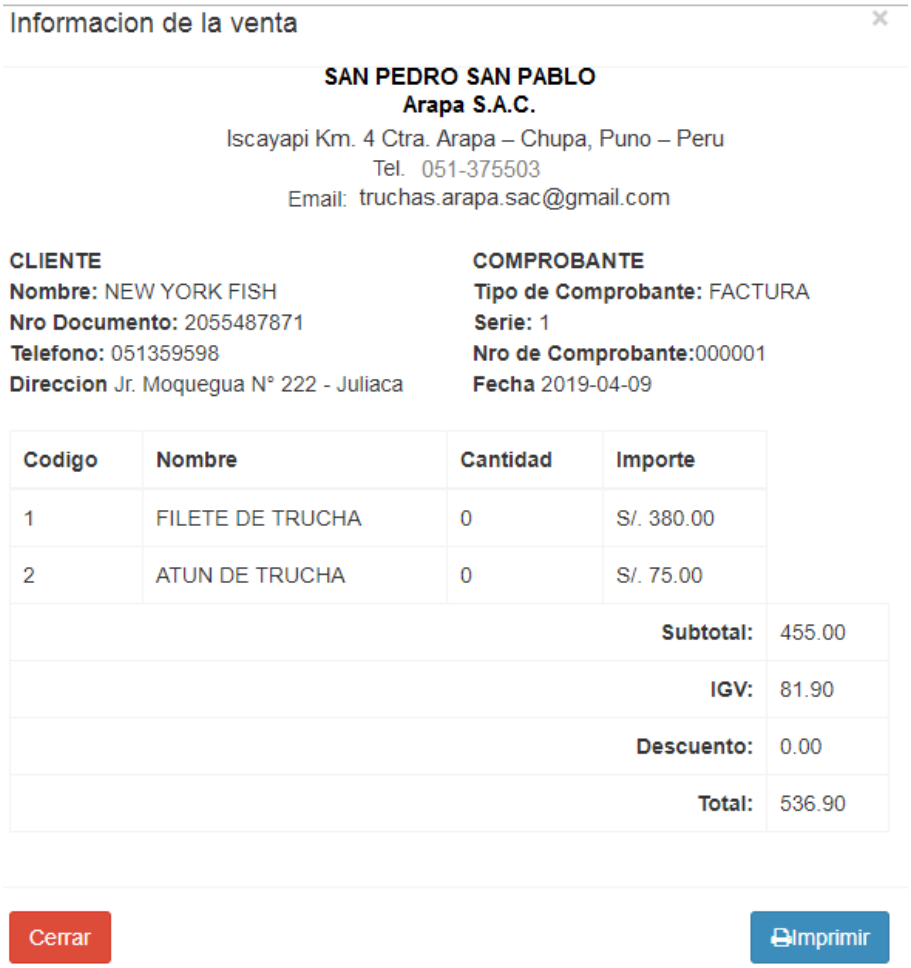


4) El sistema genera el total de las ventas realizadas detalladas



Aparte de que puedes visualizar en pantalla también se puede descargar en Excel y Pdf para que los responsables puedan archivarlos si es necesario.

5) El sistema también genera el comprobante de pago: ya sea nota de venta, boleta o factura



Informacion de la venta

SAN PEDRO SAN PABLO
Arapa S.A.C.
Iscayapi Km. 4 Ctra. Arapa – Chupa, Puno – Peru
Tel. 051-375503
Email: truchas.arapa.sac@gmail.com

CLIENTE
Nombre: NEW YORK FISH
Nro Documento: 2055487871
Telefono: 051359598
Direccion Jr. Moquegua N° 222 - Juliaca

COMPROBANTE
Tipo de Comprobante: FACTURA
Serie: 1
Nro de Comprobante:000001
Fecha 2019-04-09

Codigo	Nombre	Cantidad	Importe
1	FILETE DE TRUCHA	0	S/. 380.00
2	ATUN DE TRUCHA	0	S/. 75.00

Subtotal:	455.00
IGV:	81.90
Descuento:	0.00
Total:	536.90

Cerrar Imprimir

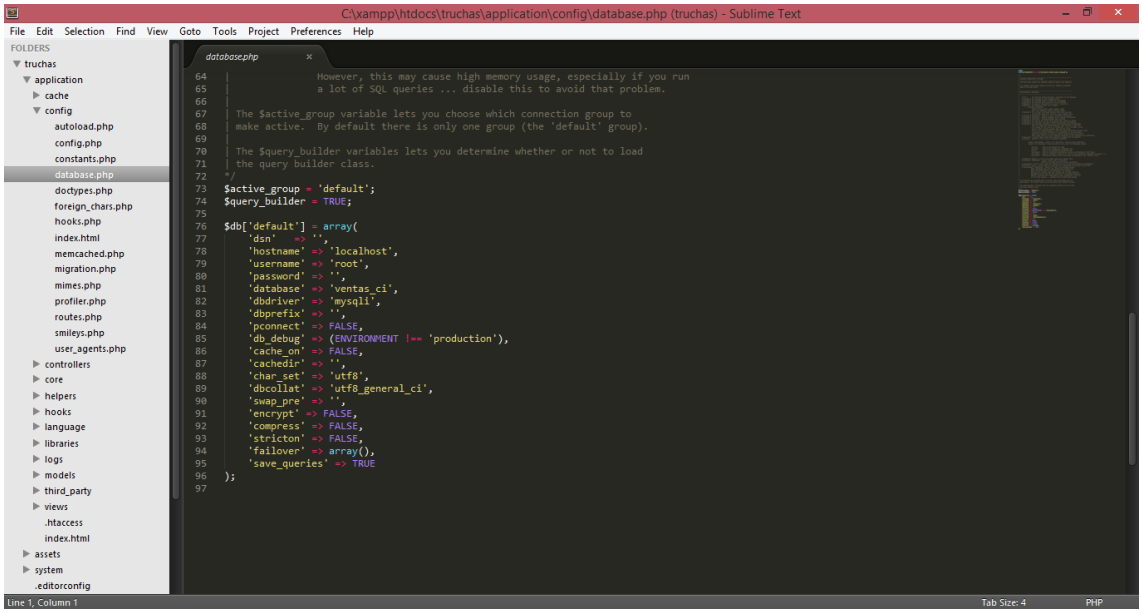
De esta forma se puede imprimir el comprobante de pago que sea necesario.

Anexo 5. Algoritmo selectivo

```
<?php foreach($productos as $producto):?>
                        <tr>
                            <?php if ($producto->stock_total%$producto->uni_caja == 0) {
                             if ($producto->uni_caja == 1) {
                               $to3=$producto->stock_total*$producto->uni_caja;
                               $to=$producto->stock_total/$producto->uni_caja;
                               $to2= "0 / ".$producto->stock_total/$producto >uni_caja;
                             }else {
                               $to3=$producto->stock_total*$producto->uni_caja;
                               $to=$producto->stock_total/$producto->uni_caja;
                               $to2= $producto->stock_total/$producto->uni_caja;
                             }
                            }else {
                             if ($producto->uni_caja == 1) {
                               $to3=$producto->stock_total*$producto->uni_caja;
                               $to=$producto->stock_total/$producto->uni_caja;
                               $to2= "0 / ".$producto->stock_total/$producto->uni_caja;
                             }else {
                               $to3=$producto->stock_total*$producto->uni_caja;
                               $to=$producto->stock_total/$producto->uni_caja;
                             $decimales = explode(".",$producto->stock_total/$producto->uni_caja);
                             $to2= $decimales[0]." / ".($producto->stock_total%$producto->uni_caja);
                             }
                            }?>
                            <?php  if($to3 > 42){?>
                             <td><?php echo $producto->nombre;?></td>
                            <td><?php echo '--> '.$to2; ?></td>
                             <?php }?>
                             </tr>
                    <?php endforeach;?>
```

Anexo 6. Código fuente

Conexión de –Base de Datos



Esta es la estructura o código fuente que se realizó en este trabajo de investigación.

Codigo fuente usurios:

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Usuarios extends CI_Controller {

  private $permisos;
        public function __construct(){
                parent::__construct();
                $this->permisos = $this->backend_lib->control();
   if (!$this->session->userdata("login")) {
                        redirect(base_url());
                }
                $this->load->model("Usuarios_model");
        }

        public function index(){
                $data = array(
    'permisos' => $this->permisos,
    'usuarios' => $this->Usuarios_model->getUsuarios()
   );
   $this->load->view("layouts/header");
   $this->load->view("layouts/aside");
   $this->load->view("admin/usuarios/list",$data);
   $this->load->view("layouts/footer");
```

```
        }
public function add()
        {
                $data = array(
  'roles' => $this->Usuarios_model->getRoles()
 );
 $this->load->view("layouts/header");
 $this->load->view("layouts/aside");
 $this->load->view("admin/usuarios/add",$data);
 $this->load->view("layouts/footer");
        }
public function store(){
                $nombres = $this->input->post("nombres");
                $apellidos = $this->input->post("apellidos");
                $telefono = $this->input->post("telefono");
                $email = $this->input->post("email");
                $usuario= $this->input->post("usuario");
                $password = $this->input->post("password");
                $rol = $this->input->post("rol");

                $this->form_validation->set_rules("nombres","Nombres del Usuario","required");
                $this->form_validation->set_rules("apellidos","Apellidos del Usuario","required");
                $this->form_validation-
>set_rules("usuario","Usuario","required|is_unique[usuarios.username]");
                $this->form_validation->set_rules("password","Contraseña","required");
                $this->form_validation->set_rules("rol","rol de Usuario","required");

 if($this->form_validation->run()){
  $data  = array(
                        'nombres' => $nombres,
                        'apellidos' => $apellidos,
                        'telefono' => $telefono,
                        'email' => $email,
                        'username' => $usuario,
                        'password' => sha1($password),
                        'rol_id' => $rol,
                        'estado' => "1"
                );

                if ($this->Usuarios_model->save($data)) {
                        redirect(base_url()."administrador/usuarios");
                }
                else{
                        $this->session->set_flashdata("error","No se pudo guardar la informacion");
                        redirect(base_url()."administrador/usuarios/add");
                }
 }
 else{
  $this->add();
 }
```

```
        }
public function edit($id){
  $data = array(
    'usuario' => $this->Usuarios_model->getUsuario($id),
    'roles' => $this->Usuarios_model->getRoles()
  );
  $this->load->view("layouts/header");
  $this->load->view("layouts/aside");
  $this->load->view("admin/usuarios/edit",$data);
  $this->load->view("layouts/footer");
}
public function view(){
  $idusuario = $this->input->post("idusuario");
  $data = array(
    'usuario' => $this->Usuarios_model->getUsuario($idusuario)
  );
  $this->load->view("admin/usuarios/view",$data);
}
public function update(){
  $idusuario = $this->input->post("idusuarios");
                $nombres = $this->input->post("nombres");
                $apellidos = $this->input->post("apellidos");
                $telefono = $this->input->post("telefono");
                $email = $this->input->post("email");
                $usuario= $this->input->post("usuario");
                $rol = $this->input->post("rol");

  $UsuarioActual = $this->Usuarios_model->getUsuario($idusuario);

     if ($usuario == $UsuarioActual->username) {
                        $is_unique = "";
                }else{
                        $is_unique= '|is_unique[clientes.num_documento]';
                }

                $this->form_validation->set_rules("nombres","Nombres del Usuario","required");
                $this->form_validation->set_rules("apellidos","Apellidos del Usuario","required");
                $this->form_validation->set_rules("usuario","Usuario","required".$is_unique);
                $this->form_validation->set_rules("rol","rol de Usuario","required");

    $data  = array(
                        'nombres' => $nombres,
                        'apellidos' => $apellidos,
                        'telefono' => $telefono,
                        'email' => $email,
                        'username' => $usuario,
                        'rol_id' => $rol
                );

                if ($this->Usuarios_model->update($idusuario,$data)) {
```

```
                redirect(base_url()."administrador/usuarios");
            }
            else{
                $this->session->set_flashdata("error","No se pudo guardar la informacion");
                redirect(base_url()."administrador/usuarios/edit".$idusuario);
            }
        }
    public function delete($id){
            $data  = array(
                'estado' => "0",
            );
            $this->Usuarios_model->update($id,$data);
            echo "administrador/usuarios";
        }
}
```