Imagen que contiene dibujo, reloj, señal

Descripción generada automáticamente

Facultad de Ingeniería

**Carrera Profesional de Ingeniería de Sistemas e Informática**

**SISTEMA DE VENTAS Y FACTURACIÓN DE LA EMPRESA SHOPTEC**

**Estudiantes:**

Huanachin Ccorahua, Jhon J.

Poma Chinchay, Shamir

Verastegui Sifuentes, Carlos Miguel

LLaja Fernandez, Juan Miguel

Sierra Jeronimo, Alberto John

**Lima – Perú**

**2021**

**ÍNDICE**

[CAPÍTULO 1 - ASPECTOS GENERALES](#_heading=h.gjdgxs)

* 1. Análisis Empresarial…………………………………………………………..
  2. Descripción del Problema…………………………………………………….
  3. [Definición de Objetivos](#_heading=h.30j0zll)

[1.3.1 Objetivo General](#_heading=h.1fob9te)

[1.3.2 Objetivos Específicos](#_heading=h.3znysh7)

* 1. [Alcances y Limitaciones](#_heading=h.2et92p0)

1.4.1 Alcances

1.4.2 Limitaciones

* 1. [Justificación](#_heading=h.tyjcwt)

1.6 [Alternativa de Solución](#_heading=h.3dy6vkm)

1.7 Estado del Arte………………………………………………………………

[CAPÍTULO 2 - MARCO TEÓRICO](#_Toc403547939)

2.1 Marco teórico de programación orientada a objetos……………………

2.2 Marco teórico de las fórmulas y/o el sistema…………………………….

[CAPÍTULO 3 - DESARROLLO DE LA SOLUCIÓN](#_Toc403547942)

3.1 Diagrama de clases

3.2 Modelamiento de base de datos

3.3 Entorno visual de los formularios

3.4 Código Fuente

[CAPÍTULO 4 - RESULTADOS](#_Toc403547944)

[4.1 Resultados de la encuesta](#_Toc403547946)

* Modelo CANVAS
* Project Charter
* Diagrama de Gant

# CAPÍTULO 1

ASPECTOS

GENERALES

* 1. Análisis empresarial:

Somos la empresa SHOPTEC encargada de desarrollar software que sirvan a diversas empresas que cuenten con el rubro de ventas en sus diferentes procesos, así como también en el ayudar en la gestión de estos, facilitando su labor frente a sus clientes y generar de esta forma valor para las empresas.

* + 1. Visión

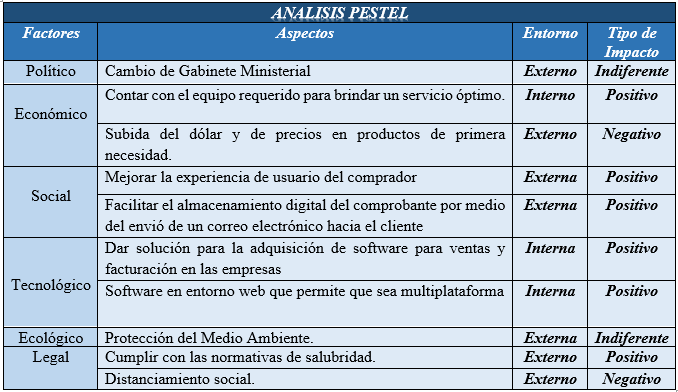
Nuestra visión como empresa es ayudar a las organizaciones a facilitar la realización de sus procesos que se realizan día a día, mediante software intuitivos y fácil de usar.

* + 1. Misión

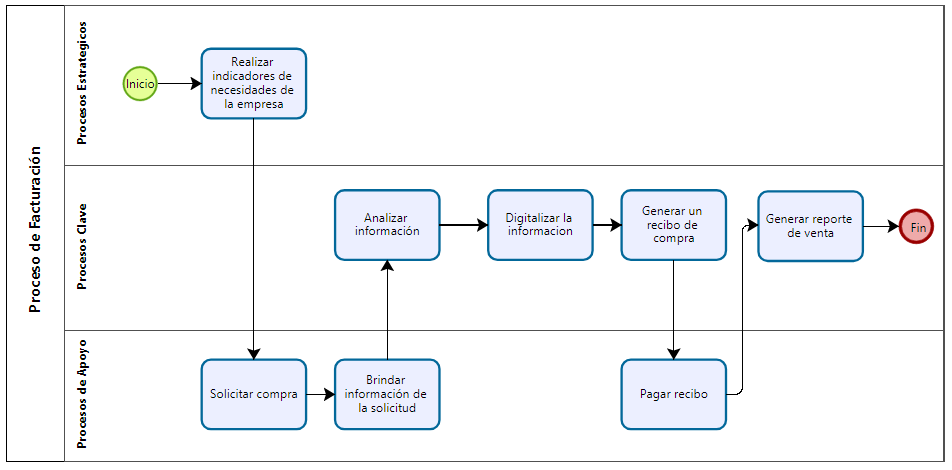
Ser la mayor organización de desarrollo de diversos programas y aplicativos que ayuden a las grandes empresas a poder optimizar en el desarrollo de los procesos que realicen.

* + 1. Análisis de entorno PESTEL

Para poder entender mejor las oportunidades y los riesgos, así como también poder evaluar el entorno tanto interno como el externo de la organización y mejorar la gestión de calidad se recurrió al análisis PESTEL sé que explicara a continuación.



* + 1. Mapa de procesos

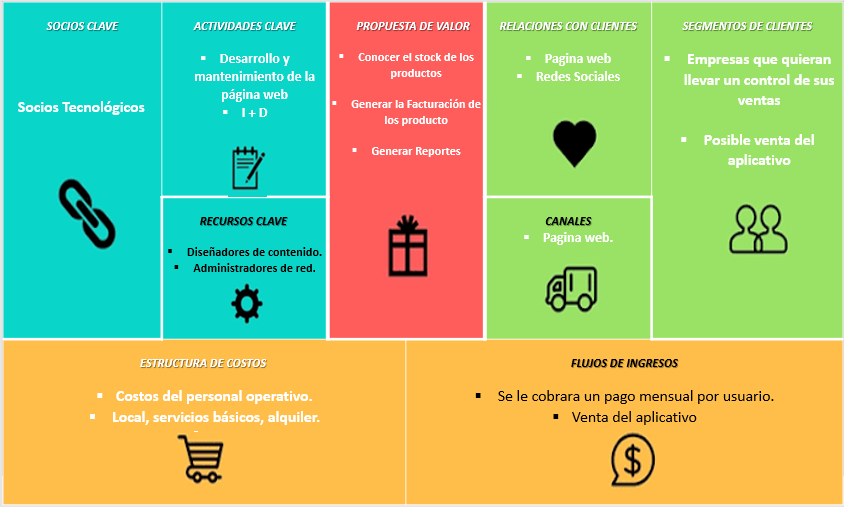


* + 1. Estrategias de procesos

Diagrama

Descripción generada automáticamente

* + 1. Modelo Canvas



* 1. Descripción del problema.

El problema se ha identificado en la falta de optimización de las pequeñas empresas o Pymes en el Perú; esto se plasma en la falta de uso de aplicativos webs para la gestión de ventas realizadas con su facturación debida.

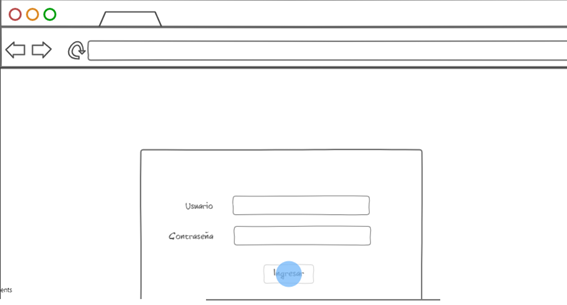
La solución que se propone es a través del planeamiento de nuestro proyecto de sistema de ventas y facturación por aplicativo web, porque es una herramienta que permite prever registros de las ventas y la facturación exacta de ellas. Asimismo, con fácil usabilidad e intuitiva para el manejo del cliente.

* 1. Definición de objetivos.
     1. Objetivo general
     + Diseñar y desarrollar un sistema web de ventas y facturación para Pymes con rubro de equipos celulares.
     1. Objetivo específico
     + Determinar la eficiencia de la aplicación de facturación y ventas.
     + Determinar el rendimiento de la aplicación de facturación y ventas.
     + Determinar si la aplicación web facilitaría el control y la emisión del comprobante de ventas.
     + Automatizar el proceso de consulta de ventas.
     + Calcular de forma eficaz y exacta la facturación.
  2. Alcances y limitaciones
     1. Alcance
     + Presentar un documento de análisis.
     + Hacer uso de una base de datos.
     + Mostrar documentos de diseño.
     + Contar con un módulo de registro de ventas.
     1. Requisitos funcionales
     + El aplicativo permitirá el registro de ventas
     + El aplicativo podrá ser utilizado en modo web.
     + El aplicativo permitirá generar la facturación de la venta.
     1. Limitaciones
     + El aplicativo solo contará con disponibilidad en rubros específicos.
     + El aplicativo solo estará disponible para los módulos de ventas y facturación.
     + El aplicativo no está diseñado para atender peticiones de otras áreas.
  3. Justificación

Actualmente en el Perú las empresas con rubro de ventas, especialmente de equipos móviles no cuentan con una aplicación para registrar sus ventas y facturación. Por este motivo y para cubrir ese servicio se desarrolló el presente proyecto.

La aplicación desarrollada da un impulso de convergencia de los servicios de ventas hacia el medio tecnológico de las aplicaciones web, aprovechando el crecimiento tecnológico en el país y su penetración en los usuarios finales.

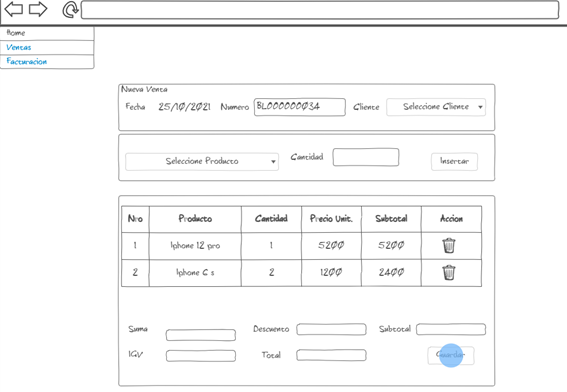
* 1. Alternativa de solución
     + Login



* + - Menú



* + - Venta



* 1. Estado del arte

En el análisis del estado del arte que aquí se realiza se agrupara en tres principales estudios: Para comenzar se hablara sobre el Desarrollo y la implementación de un sistema web para generar valor en una pyme aplicando una metodología ágil y por último se hablara sobre los Beneficios del modelo As a service en las pymes.

Investigaciones:

- DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA WEB PARA GENERAR VALOR EN UNA PYME APLICANDO UNA METODOLOGÍA ÁGIL –

Luis, C. A. P. (2016). Desarrollo e implementación de un sistema web para generar valor en una pyme aplicando una metodología ágil. Caso de estudio: Manufibras Perez SRL. UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS.

Objetivo:

En esta investigación nos mencionan e hecho de que con el paso del tiempo el uso de las tecnología en las Pymes paso de ser una opción y convertirse en un requerimiento para la supervivencia del negocio, esto debido a los múltiples beneficios que ofrece ya sea ahorrar tiempo, dinero, recursos y mejorar la calidad de servicio de sus clientes permite por consecuente el lograr generar valor, además de detallar la manera más eficiente de poder implementar estos sistemas web haciendo usos de buenas prácticas y metodologías agiles, obteniendo como principal conclusión o resultado que una empresa genera valor no con solo observar la gestión financiera, sino también otros aspectos como la innovación tecnológica y una estrategia administrativa.

<https://cybertesis.unmsm.edu.pe/bitstream/handle/20.500.12672/4668/Castillo> \_ap.pdf?sequence=1&isAllowed=y

- BENEFICIOS DEL MODELO AS A SERVICE EN LAS PYMES –

Rodríguez-Arquiñigo, J. Á. (2014). Beneficios del modelo As a service en las pymes. Interfases, 0(007), 117.

En este estudio habla sobre la importancia del modelo As a service (Como servicio en inglés), que es un modelo que puede ser usado por las pymes para poder acceder, en forma más económica, a servicios que tradicionalmente han sido costosos,  debido a la cantidad y requerimientos de los equipos involucrados, y por consiguiente mejorar la gestión de sus procesos en sus primeros años de vida.

https://revistas.ulima.edu.pe/index.php/Interfases/article/view/146

# CAPÍTULO 2

ASPECTOS

GENERALES

* 1. Marco Teórico

2.1.1. Java NetBeans

**Definición**

Es un Entorno de Desarrollo Integrado distribuido por Oracle que permite crear aplicaciones de escritorio y web mediante el uso de diferentes lenguajes de programación, su uso principal es con el lenguaje programación de Java. Es un software con licencia GPL2 que pertenece al software libre el cual puede recurrir a la interacción de APIs de NetBeans.

Por el lado del desarrollo del Back end, NetBeans posee distintos lenguajes de programación, además cuenta con librerías integradas y la posibilidad de importar externas para un uso específico. También, para el desarrollo del Front end, se apoya con un Framework que simplifica su desarrollo a través de un sistema de gestión de interfaces, librerías y herramientas.

Las versiones de NetBeans incluyen actualizaciones de los desarrolladores. Es por ello que la última versión lanzada es la de Apache NetBeans 12.3

**Clase**

Una clase es una especie de "plantilla" en la que se definen los atributos y métodos predeterminados de un tipo de objeto. Esta plantilla se crea para poder crear objetos fácilmente. Al método de crear nuevos objetos mediante la lectura y recuperación de los atributos y métodos de una clase se le conoce como instanciación.

**Herencia**

Por ejemplo, herencia de la clase C a la clase D, es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D. Por lo tanto, puede usar los mismos métodos y variables registrados como "públicos" (public) en C. Los componentes registrados como "privados" (private) también se heredan, pero se mantienen escondidos al programador y solo pueden ser accedidos a través de otros métodos públicos. Para poder acceder a un atributo u operación de una clase en cualquiera de sus subclases, pero mantenerla oculta para otras clases es necesario registrar los componentes como "protegidos" (protected), de esta manera serán visibles en C y en D pero no en otras clases.

**Objeto**

Instancia de una clase. Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos), los mismos que consecuentemente reaccionan a eventos. Se corresponden con los objetos reales del mundo que nos rodea, o con objetos internos del sistema (del programa).

**Método**

Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

**Listas**

También conocidos como vectores, son una colección de datos ordenados y generados a través de una clase, los cuales pueden ser declarados desde String hasta Objetos.

Las listas pueden implementarse mediante arreglos resultando así una estructura estática.

**Atributos**

Los atributos están asociados a las clases y objetos, estos describen la clase o el objeto de alguna manera. Un atributo puede tomar un valor definido de un dominio, o sea un rango de valores probables a tomar. Podemos decir que un atributo es un campo que guardará un valor específico. A continuación, se muestra atributos para la clase Persona.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Dominio**

Son todos los valore que puede contener un atributo, por ejemplo, un dominio para profesión puede ser: Ingeniero, Médico, Enfermera, etc. En situaciones más complejas un dominio puede ser un conjunto de clases.

**2.1.2. Java Swing:**

Java Swing es una herramienta de widgets de la interfaz gráfica de usuario (GUI) ligera de Java que incluye un amplio conjunto de widgets.  
 Es parte de Java Foundation Classes (JFC) e incluye varios paquetes para desarrollar aplicaciones de escritorio ricas en Java.  
 Swing incluye controles integrados como árboles, botones de imagen, paneles con pestañas, controles deslizantes, barras de herramientas, selectores de color, tablas y áreas de texto para mostrar el formato de texto enriquecido (RTF) o HTTP. Los componentes Swing están escritos completamente en Java y, por lo tanto, son independientes de la plataforma.

Swing ofrece la personalización de la apariencia de cada componente en una aplicación sin realizar cambios significativos en el código de la aplicación.  
 También incluye una función de apariencia y tacto conectable, que le permite emular la apariencia de los componentes nativos a la vez que tiene la ventaja de la independencia de la plataforma. Esta característica particular hace que escribir aplicaciones en Swing sea fácil y lo distingue de otros programas nativos.  
 Originalmente, la biblioteca de gráficos para Java, desarrollada por Netscape Communication Corporation, se llamaba Internet Foundation Classes (IFC). La primera versión de IFC fue el 16 de diciembre de 1996. La evolución de JFC se remonta a 1997, cuando Sun Microsystems y Netscape Communication Corporation tuvieron la idea de fusionar IFC con otras tecnologías.

**Componentes de Java Swing**

Las clases cuyo nombre comienza por J forman parte de Swing.Todas las demás están incluidas en AWT (Abstract Window Toolkit) Component es una clase abstracta que representa a cualquier componente con representación gráfica.

Container es un componente que puede contener a otros componentes gráficos (p.ej. JPanel). JFrame permite representar ventanas, si bien también se pueden utilizar clases como JApplet o JDialog.

Los frames (como JFrame) son contenedores, por lo que incluyen un “panel de contenido” (content pane) al cual se le pueden añadir componentes gráficos (etiquetas, botones, cajas de texto, etc.) y otros contenedores (como paneles JPanel). Las interfaces gráficas de usuario se construyen con componentes, cada uno de los cuales está preparado para responder a distintos tipos de eventos. Algunos de los componentes incluidos en Swing son:

JLabel (etiqueta para mostrar texto)

JTextBox & JTextArea (cajas de texto para la entrada de datos)

JButton (botón)

JCheckBox (caja de comprobación, para elegir opciones)

JRadioButton (para elegir opciones mutuamente excluyentes)

JList (lista de opciones)

JComboBox (lista desplegable de opciones)

JScrollBar (barra de scroll)

JTree (árbol)

JTable (tabla)

Menús: JMenuBar, JMenu, JMenuItem

Ventanas de diálogo: JOptionPane

Ventanas estándar: JFileChooser, JColorChooser

Distribución del espacio (layout management)

Antes de construir una interfaz gráfica de usuario es importante saber cómo se distribuyen especialmente los componentes. Los layout managers controlan la forma en la que colocan los componentes dentro de un contenedor: FlowLayout (por defecto para un JPanel) coloca los componentes de izquierda a derecha, de arriba abajo. Los componentes se van colocando a la derecha de los ya existentes hasta que se añade una nueva fila cuando no queda espacio suficiente en la fila actual. Cuando el contenedor se redimensiona, los componentes se redistribuyen automáticamente.

BorderLayout (por defecto para un JFrame) permite dividir el espacio disponible de la siguiente forma:

**NORTH**

**WEST CENTER EAST**

**SOUTH**

Usualmente, al JFrame se le añaden paneles usando BorderLayout y luego se rellenan los paneles JPanel usando FlowLayout Existen otros layout managers más sofisticados (como GridLayout o GridBagLayout) e incluso podemos crear nuestros propios gestores de layout si implementamos las interfaces java.awt.LayoutManager y java.awt.LayoutManager2.

Los paneles (como JFrame) son contenedores que pueden contener otros componentes y, además, tienen una superficie sobre la que se puede dibujar

**Relaciones entre clases**

Cuando se diseña un programa se realizan diversos artefactos que nos permiten identficar sus componentes y funcionamiento. Entre ellos tenemos a los diagramas de clases y objetos.  
 En los lenguajes orientados a objetos como JAVA es fundamental comprender como las clases existentes se relacionan. UML permite definir tres tipos de relaciones fundamentales que son:

* Asociación
* Agregación
* Composición

Notación de relación entre las clases

Tienen las siguientes partes:

* Nombre de la relación (un verbo).
* La ocurrencia de cada clase (rangos ubicados en los extremos).
* El rol de cada clase en la relación.

La multiplicidad es el número de instancias de una clase que se relacionan con una instancia de otra clase.

|  |  |
| --- | --- |
| **Multiplicidad** | **Simbología** |
| **Uno y solo uno** | 1 |
| **Cero o uno** | 0 .. 1 |
| **Desde "N" hasta "M"** | N .. M |
| **Varios** | \* |
| **Cero a varios** | 0 .. \* |
| **Uno a varios** | 1 .. \* |

**Asociación:**

Es una relación entre dos clases que requieren relacionarse (como una orden y un cliente), mediante UML y el diagrama de clases de representa mediante una línea y en algunos casos incluye una flecha siguiendo la estructura de una oración. Esta flecha apunta al objeto y el otro extremo representa al sujeto.

Relación unidireccional:

[Diagrama

Descripción generada automáticamente](https://1.bp.blogspot.com/-BslNX2cTtcE/XvvGOR-pHWI/AAAAAAAAGHM/EUorWZA1prc6f6c1HZj82iEWyzb7yB-lACLcBGAsYHQ/s1600/relaciones01.png)

Relación bidireccional:

[Diagrama

Descripción generada automáticamente](https://1.bp.blogspot.com/--InPt2ty3cg/Xv58E-fX3TI/AAAAAAAAGIo/oR2rlfEzOn4lzjcTxv3eFi8lLt5qJfatACLcBGAsYHQ/s1600/relaciones02.png)

**Agregación:**

Es un tipo especial de asociación entre clases conocido como: “contiene a” o “es contenido en”. Ambas clases tiene una vida independiente, pero una de estas (llamada invitado) trabaja en orden a otra (huésped). Esto le permite a huésped usar características de invitado en la ejecución de algunas tareas.  
  
 Se representa con una línea con rombo blanco, donde el rombo está en el huésped y el otro lado en el invitado.

[Diagrama

Descripción generada automáticamente](https://1.bp.blogspot.com/-cMN0Y6Fhh9E/Xvv_Y5dlGOI/AAAAAAAAGHg/AiwXl7uYE_cBEHu07ktLUn7t2fEiISpHgCLcBGAsYHQ/s1600/relaciones03.png)

**Composición de clases:**

Tipo especial de agregación. Conocida como "es parte de" o "es un todo de". En este caso las dos partes necesitan de ellas para existir (una no existe sin la otra), de manera que existe una clase (todo) que utiliza características de otra (parte) para la ejecución de alguna tarea.  
 Se representa con un rombo relleno, que está junto al "todo" y al otro extremo la "parte".

[Diagrama

Descripción generada automáticamente](https://1.bp.blogspot.com/-_RZU98KS5Eo/XvwAsKv6dfI/AAAAAAAAGHs/oIrU1ooxATQHXU45PD5EXKwSHElx7gsVQCLcBGAsYHQ/s1600/relaciones04.png)

**2.1.3. POO**

La programación orientada a objetos es un paradigma de programación en el que los programas se ven como formados por entidades llamadas objetos que recuerdan su propio estado interno y que se comunican entre sí mediante el paso de mensajes que se intercambian con la finalidad de cambiar sus estados internos, compartir información y solicitar a otros objetos el procesamiento de dicha información.

**Los 4 pilares de la Programación Orientada a Objetos**

Existen muchos conceptos en programación orientada a objetos, como clases y objetos, sin embargo, en el desarrollo de software con programación orientada a objetos, existen un conjunto de ideas fundamentales que forman los cimientos del desarrollo de software. A estos 4 conceptos que vamos a ver les llamamos los 4 pilares de la programación orientada a objetos.

Esto no quiere decir que fuera de estos 4 pilares no existan otras ideas igual de importantes, sin embargo, estos 4 pilares representan la base de ideas más avanzadas, por lo que es crucial entenderlos.

Estos pilares son: abstracción, encapsulamiento, herencia y polimorfismo.

|  |  |
| --- | --- |
|  | public class Carro  {       public string Marca;       public int AñoSalidaAlMercado { get; set; }       public void Acelerar()      {      }  } |
|  |  |

**Abstracción**

De acuerdo a la RAE, una de las acepciones de abstraer es:

“Hacer caso omiso de algo, o dejarlo a un lado.” Y ofrece como ejemplo: “Centremos la atención en lo esencial abstrayendo DE consideraciones marginales.”

El ejemplo dado captura la esencia del concepto de abstraer. Cuando hacemos una abstracción, queremos omitir detalles que no son necesarios para nosotros, y queremos solamente mostrar lo que sí es relevante.

Desde el punto de vista del desarrollo de software, podemos ver que con una clase podemos realizar una abstracción de una entidad del mundo real. Tomemos por ejemplo la clase carro que hicimos, esta tiene la posibilidad de guardar datos relacionados a la marca y al año de salida al mercado del carro, pero, ¿Por qué solamente estas dos informaciones? Un carro del mundo real tiene más propiedades, como el color y el modelo. Sin embargo, debemos preguntarnos, ¿Son estas informaciones relevantes para nuestro software?

Nuestra clase abstrae todo lo que representa un carro, tomando solamente lo que nos interesa, descartando todo lo demás.

**Encapsulamiento:**

Ya sabes que puedes utilizar clases para modelar entidades las cuales son relevantes para tu aplicación, sabes además que puedes guardar datos dentro de objetos, y también ejecutar funcionalidad. La pregunta que debemos hacernos es, ¿Debe cualquiera poder modificar de manera directa estos datos? ¿Debe cualquiera ejecutar cualquier funcionalidad de nuestros objetos en cualquier momento? Normalmente esto no es algo que queremos, nosotros queremos poder controlar la manera en que se asignen los datos, queremos poder controlar quién ve la data interna de nuestros objetos, e incluso quizás queramos controlar la ejecución de funcionalidad de nuestros objetos. Para esto tenemos el concepto de encapsulamiento.

El encapsulamiento nos permite controlar quien puede ver y utilizar los distintos módulos internos de nuestro sistema. En términos de clases, con el encapsulamiento definimos el acceso a los miembros de la clase.

En C# podemos utilizar modificadores de acceso para definir el control de agentes externos a distintas partes de nuestro sistema, como clases, miembros de las clases, interfaces, entre otros. Supongamos que tenemos una variable, llamada velocidad, la cual queremos colocar en nuestra clase Carro, para indicar la velocidad en la cual se desplaza un vehículo en particular. Sin embargo, queremos que solamente dentro de la clase podamos ver y modificar el valor de dicha variable. Esto lo podemos hacer o con un campo o con una propiedad. Hagámoslo con una propiedad:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | public class Carro  {        public string Marca;        public int AñoSalidaAlMercado { get; set; }        private int Velocidad { get; set; }        public void Acelerar()      {          Velocidad += 10;      }    } |

Cuando hagamos una instancia de la clase Carro, no podremos acceder al valor de la propiedad Velocidad, ni tampoco podemos alterarlo desde afuera. Lo que sí podemos hacer es utilizar la función acelerar para aumentar el valor de la velocidad en 10 unidades. Esta es una de las ventajas del encapsulamiento: Nos permite controlar la manera en que se va a alterar la data interna de nuestro objeto.

Si quisiéramos que agentes externos puedan ver el valor la propiedad Velocidad, pero que no puedan alterar libremente dicho valor, podemos utilizar la siguiente sintaxis:

|  |  |
| --- | --- |
| 1 | public int Velocidad { get; private set; } |

**Herencia**

Compartir código es una importante y crucial característica de cualquier proyecto de software. Compartir código permite ahorrar trabajo cuando queremos hacer un cambio en nuestro sistema; permite que un solo algoritmo pueda procesar distintas clases de entidades; entre otras cosas.

Hay varias maneras de compartir código, una de ellas es utilizando herencia. La herencia es una relación especial entre dos clases, la clase base y la clase derivada, en donde la clase derivada obtiene la habilidad de utilizar ciertas propiedades y funcionalidades de la clase base, incluso pudiendo sustituir funcionalidad de la clase base. La idea es que la clase derivada “hereda” algunas de las características de la clase base.

Podemos ver un ejemplo de la clase Carro. Un carro es un tipo de vehículo, además, queremos procesar otro tipo de vehículos, cada uno con su entidad, como camión. Un carro y un camión comparten el concepto de velocidad, además, ambos tienen la capacidad de acelerar, y ambos tienen la capacidad de ir de reversa, sin embargo, cuando un camión va de reversa, este debe emitir un sonido. Finalmente, un carro debe poder encender la radio. Vamos entonces a modelar esto:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | public class Vehículo  {        public string Marca;        public int AñoSalidaAlMercado { get; set; }        public int Velocidad { get; private set; }        public void Acelerar()      {          Velocidad += 10;      }        public virtual void Reversa()      {            Console.WriteLine("Voy de reversa!");      }  }    public class Carro: Vehículo  {       public void EncenderRadio()       {         Console.WriteLine("Encendiendo la radio");     }  }    public class Camión: Vehículo  {      public override void Reversa()        {          base.Reversa();          Console.WriteLine("BEEP BEEP BEEP!");      }  } |
|  |  |

Vemos que tenemos 3 clases: Vehículo, Carro y Camión. Carro y Camión heredan de la clase Vehículo. La relación de herencia se representa de esta manera:

|  |  |
| --- | --- |
| 1 | class Carro: Vehículo |

Con esta sintaxis decimos que Carro es una clase derivada de Vehículo.

Vemos además que la función Acelerar está definida en la clase Vehículo, esto hace que todas las clases derivadas pueden hacer uso de dicha función. Lo mismo sucede con los campos y propiedades.

Ciertamente las clases Carro y Camión pueden definir sus propios miembros que no se relacionan con la clase Vehículo. Por ejemplo, la clase Carro tiene el método EncenderRadio el cual solo esta lo tiene.

Podemos también modificar funcionalidad de la clase base. Para esto, en la clase base, el método debe estar marcado como virtual. Y cuando se quiera sobrescribir, es decir, cambiar o agregar funcionalidad, esto lo podemos hacer haciendo un override, tal cual vemos en la clase Camión. Dentro del método Reversa de la clase Camión, tenemos el código base.Reversa(); el cual sirve para invocar el método reversa de la clase base.

Podemos utilizar el código anterior de la siguiente manera:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | Carro miCarro = new Carro();    miCarro.AñoSalidaAlMercado = 2018;    miCarro.Acelerar();    Console.WriteLine(miCarro.Velocidad);    miCarro.Reversa();    Console.WriteLine("-------");    Camión miCamion = new Camión();    miCamion.Acelerar();    miCamion.AñoSalidaAlMercado = 2012;    miCamion.Reversa(); |

**Clases Abstractas:**

¿Qué tal si quisiéramos que la clase Vehículo no pudiera ser instanciada? Podemos marcarla como una clase abstracta. Una clase abstracta es aquella que no puede ser instanciada. Es útil en situaciones de herencia donde no queremos que los usuarios instancien la clase base, sino que queremos que instancien solamente las clases derivadas. Para marcar la clase Vehículo como abstracta utilizamos abstract:

|  |  |
| --- | --- |
| 1 | public abstract class Vehículo |

¿Qué tal si quisiéramos obligar a las clases derivadas a implementar una función específica, sin que la clase base dé una implementación por defecto? Para esto podemos marcar el método como abstract. Ejemplo:

|  |  |
| --- | --- |
| 1 | public abstract void MetodoObligatorio(); |

**Interfaces:**

Las interfaces nos ayudan a realizar otro tipo de herencia. Mientras que una clase base nos ofrece implementación por defecto de algunos métodos, como el método reversa de la clase Vehículo, las interfaces nos ofrecen un conjunto de miembros que las clases que implementan la interfaz deben implementar. Las interfaces no pueden ser instanciadas, igual que las clases abstractas.

Históricamente, una diferencia fundamental entre interfaces y clases abstractas es que las clases abstractas nos permiten crear implementaciones por defecto de métodos y las interfaces no. Sin embargo, es posible que en C# 8 eso cambie con la introducción de implementaciones por defecto en interfaces.

Nota: Aunque las interfaces son un tipo de herencia, es normal referirse a herencia solamente al caso en el que tenemos una clase base.

**Polimorfismo:**

Cuando empezamos a hablar de herencia, dijimos que la herencia “permite que un solo algoritmo pueda procesar distintas clases de entidades”. La idea es que podemos tener una función la cual reciba un parámetro, como una clase base, y podemos pasarle a ese método objetos que sean instancias de las clases derivadas de dicha clase base. Lo mismo ocurre si el método recibe como parámetro una interfaz. Podemos pasarle a dicho método cualquier clase que implemente dicha interfaz.

Polimorfismo significa de muchas formas. En nuestro caso llamamos polimorfismo cuando un método recibe un parámetro que abarca varios tipos.

Veamos un ejemplo de polimorfismo donde pasamos a un método la clase base Vehículo:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | static void Reparar(Vehículo vehículo)    {        Console.WriteLine("Iniciando reparación");        Console.WriteLine("Probando acelerador");        Console.WriteLine($"Velocidad inicial {vehículo.Velocidad}");        vehículo.Acelerar();        Console.WriteLine($"Velocidad final {vehículo.Velocidad}");        Console.WriteLine("Probando reversa");        vehículo.Reversa();        Console.WriteLine("Listo!");    } |

Este método invoca los métodos Acelerar y Reversa del vehículo que se le envíe como parámetro. La ventaja que esto ofrece es que podemos generalizar algoritmos para que funcionen con distintos tipos. En este caso, este método va a funciona con cualquier clase que herede de Vehículo, en tal sentido, incluso si en el futuro agregamos la clase Motocicleta, la cual hereda de Vehículo, podemos utilizar esta nueva clase con el método Reparar, y va a funcionar perfectamente. De esta manera se da el polimorfismo, pues el método reparar puede trabajar con varios tipos distintos.

En el método reparar no podemos hacer uso del método EncenderRadio de la clase Carro, pues la clase Vehículo no implementa dicho método. Lo que podríamos hacer es utilizar el operador is para castear a Carro en caso de que el Vehículo sea un carro:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | if (vehículo is Carro miCarro)    {        miCarro.EncenderRadio();    } |

Esta sintaxis es una manera resumida de decir:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | if (vehículo is Carro)    {        Carro miCarro = vehículo as Carro;        miCarro.EncenderRadio();    } |

**Framework en Java**

Los JSP (JavaServer Pages), permite insertar fácilmente código Java dentro de la página HTML. Con esta Solución, las aplicaciones Web adoptaron a los JSP como figura central, lo que pronto traería como consecuencia problemas con el control del flujo, así como en el mantenimiento de páginas con demasiado código web.

**Bootstrap**

Bootstrap fue creado por Mark Otto y Jacbod Thornton ambos desarrolladores de Twitter, ellos buscaban un marco de trabajo para fomentar la consistencia entre las herramientas internas, las cuales llevaban a inconsistencias y a una gran carga de trabajo en su mantenimiento.

El primer desarrollo en condiciones reales ocurrió durante la primera “Semana de Hackeo” (Hackweek) de Twitter. Mark Otto mostró a algunos colegas como acelerar el desarrollo de sus proyectos con la ayuda de la herramienta de trabajo. Como resultado, decenas de temas se han introducido en el marco de trabajo. En agosto del 2011, Twitter liberó a Bootstrap como código abierto. En febrero del 2012, se convirtió en el proyecto de desarrollo más popular de GitHub.

**CSS**

Cascading Style Sheets, significa Hojas de Estilo en Cascada. Es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML. MySQL Server

MySQL es un sistema de gestión de base de datos. Una base de datos es una colección estructurada de datos. Puede ser cualquier cosa, desde una simple lista de la compra a una galería de imágenes o las grandes cantidades de información en una red corporativa. Para agregar, el acceso y proceso de los datos almacenados en una base de datos, se necesita un sistema de gestión de base de datos como MySQL Server.

# CAPÍTULO 3

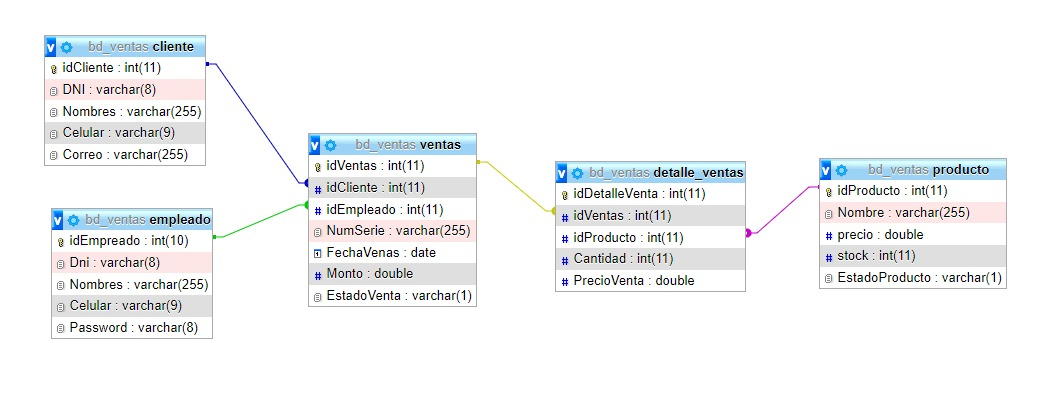
DESARROLLO DE LA SOLUCION

* 1. **Diagrama de Clases**

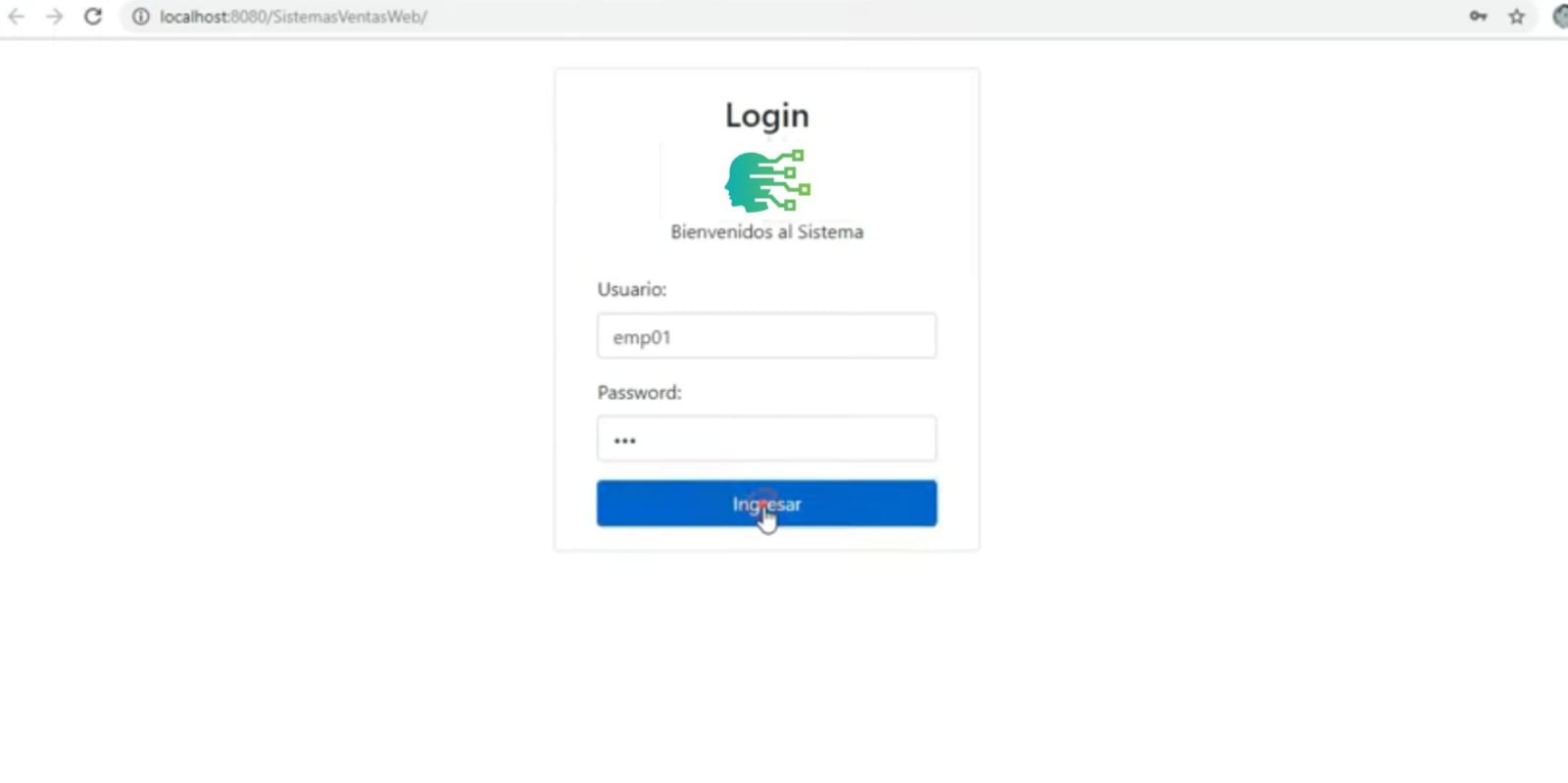
Diagrama

Descripción generada automáticamente

* 1. **Modelamiento de base de datos**

****

* 1. **Entorno visual de los formularios**
* La aplicación web podrá proporcionar una ventana de inicio( login ), en el cual para ingresar se pondrá el usuario y password.
* La aplicación web contara con una ventana, donde figura los datos del cliente, del producto y la función para buscar y eliminar productos. Asimismo, se podrá modificar y eliminar productos, con dos botones finales de generar venta y cancelar.
* La aplicación web al finalizar el registro de venta, obtendremos una facturación de toda la venta.
* Ventana de Login



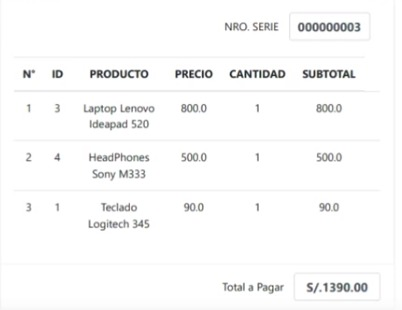
* Barra de menú del aplicativo web

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

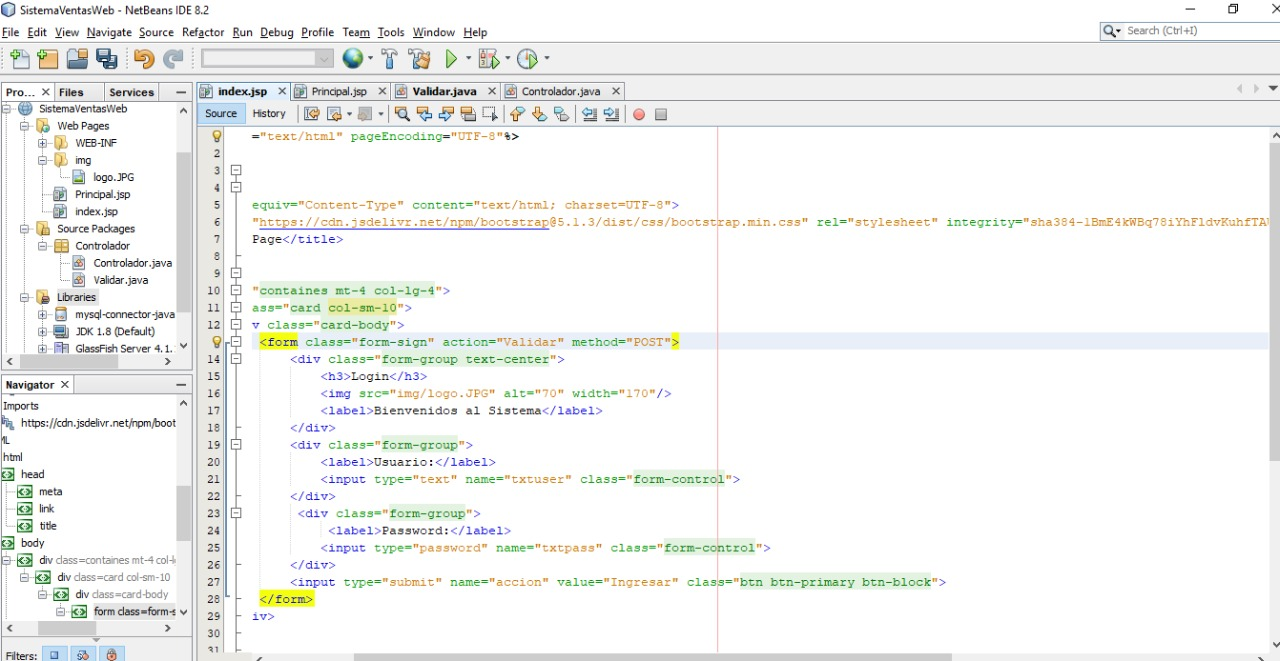
Descripción generada automáticamente

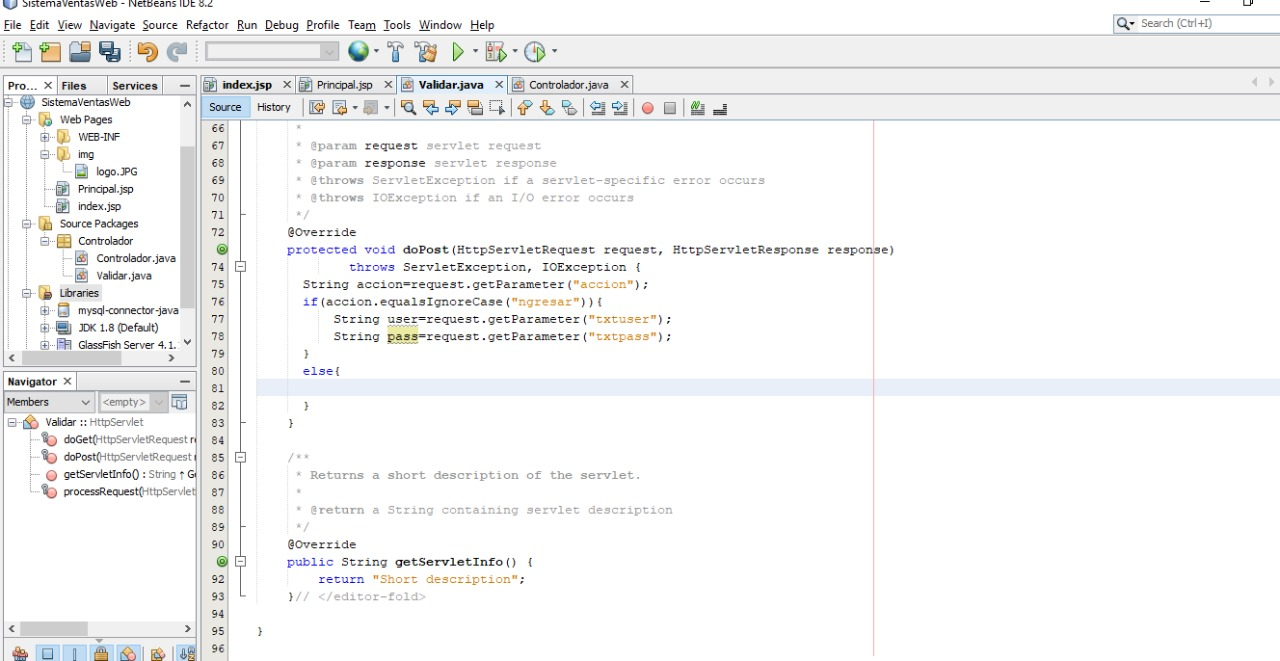
* Interfaz de usuario gráfica, Aplicación

  Descripción generada automáticamenteVentana de registro de venta de productos
* Ventana de facturación



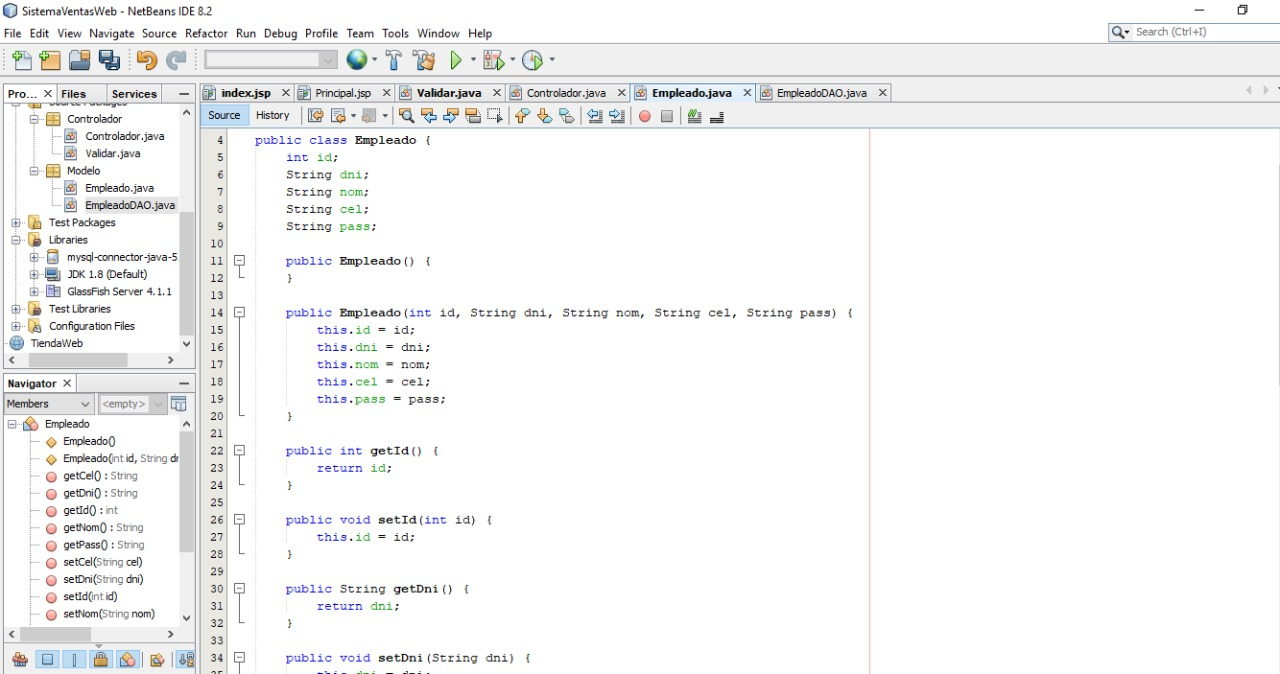
* 1. **Código Fuente**





Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

******

# CAPÍTULO 4

RESULTADOS

4.1 Resultados

Calendario

Descripción generada automáticamente

4.1.1 Resultados de la encuesta

Para el análisis de los resultados recaudados por el uso del Aplicativo web, se ha proporcionado una encuesta al usuario de la empresa Mulcotec donde se obtendrán los datos de las frecuencias y porcentajes de su eficiencia. De mismo modo, para medir la experiencia del aplicativo se pidió se logeen e interactúen realizando ventas y emitiendo la boleta electrónica de la venta, utilizando las ventanas de uso correspondiente, junto con la aplicación para tener un resultado del tiempo que tarda el usuario en efectuar las operaciones. Por último, se pedirá al usuario que deje su valoración de la interfaz y del fácil uso del aplicativo web.

# **La población**

La población de la investigación está compuesta por personal de área de ventas en la empresa Mulcotec.

# **La muestra**

La investigación está sujeta como representación significativa, una muestra de 20 empleados que laboran en la empresa Mulcotec.

# **Las encuestas**

Como método fundamental tenemos una encuesta de satisfacción realizada hacia 20 empleados en la cual se dio a conocer el modo de pensar con respecto a la experiencia, eficiencia del aplicativo y valoración de la interfaz del aplicativo web.

# **Las técnicas de procesamiento de análisis de datos**

Las tablas estadísticas: Se muestra los resultados en tablas acerca de la cantidad y el porcentaje de las encuestas realizadas.

Los gráficos estadísticos: Esto permitirá ver los distintos gráficos. Es la representación gráfica de los resultados de las encuetas estadísticas.

**Análisis de la encuesta aplicada**

Por facilidad de uso:

1. Aprender a utilizar el aplicativo web fue fácil
2. Fue fácil que el aplicativo web muestre la información de la venta.
3. Fue fácil de interactuar con el aplicativo web

**Objetivo General**

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

Fuente: Elaboración propia

Análisis:

Según los resultados de la encuesta se observa que el 70% de los empleados se encuentran totalmente de acuerdo y solo el 30% están de acuerdo.

Por Utilidad:

1. El uso del aplicativo web permitió consulta la

venta rápidamente.

1. E l uso del aplicativo web redujo el tiempo

de registros y consultas de las ventas.

1. El uso del aplicativo web permitió obtener el registro de las ventas.

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

Fuente: Elaboración propia

Análisis:

Según los resultados de la encuesta se observa que el 78.3% de los empleados se encuentran totalmente de acuerdo y solo el 21.7% están de acuerdo.

Por Usabilidad:

1. Los componentes visuales del aplicativo web son ordenados.
2. Los componentes visuales del aplicativo web son interactivos.
3. Los componentes visuales del aplicativo web son complejos.

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

Fuente: Elaboración Propia

Análisis:

Según los resultados de la encuesta se observa que el 41% de los empleados están de acuerdo, el 25% está totalmente de acuerdo y los siguientes resultados de 10%, 8.3%, 15% son obtenidos de acuerdo con la pregunta 9 solamente.

Intensión de Uso:

1. Tengo la intensión de usar el aplicativo web con frecuencia para el registro de ventas y obtencion de la boleta electronica.

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

Fuente: Elaboración Propia

**Anexos**

**PROYECT CHARTER:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CONTROL DE VERSIONES** | | | | | |
| ***Versión*** | ***Hecha por*** | ***Revisada por*** | ***Aprobada por*** | ***Fecha*** | ***Motivo*** |
| 1.0 | Grupo 01 | JHCC | AJS | 06-09-2021 | Elaboración de Documento |
| 1.0.1 | Grupo 01 | JHCC | AJS | 13-09-2021 | Corrección de Documento |

*PROJECT CHARTER*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre del Proyecto** | | **Siglas del Proyecto** | | | |
| **Sistema de venta y Facturación de la empresa SHOPTEC** | | **V.F.E.S** | | | |
| **Descripción del Proyecto:** | | | | | |
| **Desarrollo de una aplicación web con el que se dará a conocer el índice de masa corporal de una persona y que tan saludable se encuentras.**  **Integrantes:**  Huanachin Ccorahua, Jhon J.  LLaja Fernandez, Juan Miguel  Poma Chinchay, Shamir  Sierra Jeronimo, Alberto John  Verastegui Sifuentes, Carlos Miguel  **Para el análisis y diseño se utilizará la metodología RUP y para la programación se utilizará el lenguaje de programación de Java y como Ide NetBeans, además de un la utilización de principios básicos de la Programación Orientada a Objetos (POO).**  **El proyecto tendrá como fecha de inicio 09-08-2021 y como fecha fin 06-09-2021.**  **El proyecto será desarrollado en las instalaciones de la empresa.** | | | | | |
| **Definición del Producto del Proyecto:** | | | | | |
| **El proyecto consiste en el desarrollo de la documentación de análisis, la documentación del diseño, el código fuente de la aplicación, los informes de pruebas y la implementación en un ambiente de producción.**  **La aplicación implementada deberá tener la siguiente funcionalidad**  **Ingreso de datos para la medición del Índice de Masa Corporal** | | | | | |
| **Definición de Requisitos del Proyecto:** | | | | | | |
| **Requisitos Funcionales:**  El sistema deberá permitir el ingreso de datos para obtener un resultado.  **Requisitos No funcionales:**  -Rendimiento  -Funcionalidad  -Accesibilidad  -Usabilidad  -Eficiencia  -Seguridad  -Eficiencia | | | | | | |
| **Objetivos del Proyecto:** | | | | | | | | | |
| ***Concepto*** | | ***Objetivos*** | | | ***criterio de Éxito*** | | | | |
| ***1. Alcance*** | | La aplicación de escritorio que calcula el índice de masa corporal, la aplicación está hecha con el lenguaje de programación de Java puede ser utilizada en pc, la persona tendrá que ingresar unos datos y después de hacer clic en el botón “Calcular” saldrá el resultado en la página web | | | 100% funcional, puede ser utilizada en pc, aceptados por el cliente | | | | |
| ***2. Tiempo*** | | Del 09-08-2021 al 06-09-2021 | | | Cumplimiento con la fecha programada | | | | |
| ***3. Costo*** | | S/. 12 500.00 soles | | | Cumplimiento con presupuesto establecido | | | | |
| **Finalidad del Proyecto:** | | | | | | | | |
| **Es proporcionar al cliente su índice de masa corporal para que pueda tomar una decisión de inscribirse en los servicios del gimnasio.** | | | | | | | | |
| **Justificación del Proyecto:** | | | | | | | | |
| ***Justificación Cualitativa*** | | | | ***Justificación Cuantitativa*** | | | | |
| Desarrollar nuevas destrezas o habilidades | | | | *Flujo de Ingresos* | | *Contrato con el Cliente* | | |
| Generar innovación y desarrollo en el contexto peruano | | | | *Flujo de Egresos* | | Alquiler de local  Pago de Servicios  Compra de Equipos de Computo  Pago de Planilla | | |
| Contribuir a mejorar la salud de las personas brindando un servicio para que puedan saber su índice de masa corporal | | | | *VAN* | | S/. 24930.56 soles | | |
|  | | | | *TIR* | | 169 % | | |
|  | | | | *RBC* | | S/ 1.50 soles | | |

|  |  |  |
| --- | --- | --- |
| **Designación del Project Manager del Proyecto.** | | |
| ***Nombre*** | **Huanachin Ccorahua, Jhon J.** | ***Niveles de autoridad*** |
| ***Reporta a*** |  |  |
| ***Supervisa a*** |  |

|  |  |
| --- | --- |
| **Cronograma de Hitos del Proyecto.** | |
| ***Hito o Evento Significativo*** | ***Fecha Programada*** |
| Inicio del Proyecto | 09-08-2021 (Semana 1) |
| Avance del Proyecto | 06-09-2021 (Semana 4) |
| Presentación final de Proyecto | 06-10-2021 (Semana 9) |

|  |  |
| --- | --- |
| **Organizaciones o Grupos Organizacionales que intervienen en el Proyecto.** | |
| ***Organización o Grupo Organizacional*** | ***Rol que desempeña*** |
| Organización GYM S.A. | Es un gimnasio de entrenamiento en la cual se hace el servicio de entrenamiento personalizado, ya sea para desarrollar masa muscular o en la cual se quiera reducir de peso. |

|  |
| --- |
| **Principales Amenazas del Proyecto** |
| La persona tiene que tener internet para poder utilizar la aplicación |
| Uso indispensable de una pc. |

|  |
| --- |
| **Principales Oportunidades del Proyecto** |
| Necesidad de las personas de obtener el resultado sin salir de su casa |
| Entorno amigable de uso |
| Dato requerido por profesionales de la salud para obtener un perfil de salud del paciente |

|  |  |
| --- | --- |
| **Presupuesto Preliminar del Proyecto.** | |
| ***Concepto*** | ***Monto*** |
| Alquiler de oficina | S/. 2000.00 soles al mes |
| Pago de Servicios | S/. 500.00 soles al mes |
| Compra de Equipos de Computo | S/. 10000.00 soles al mes |
| Programador en Java | S/. 4000.00 soles al mes |
| Jefe de Proyecto | S/. 5000.00 soles al mes |

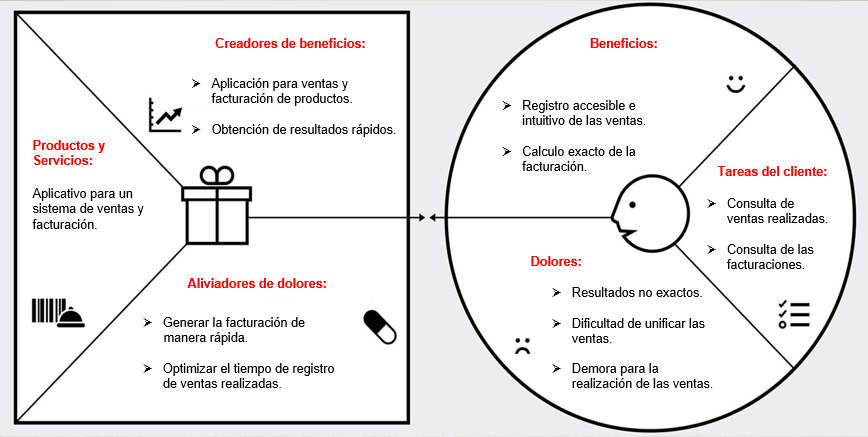
|  |  |  |  |
| --- | --- | --- | --- |
| **sponsor que autoriza el Proyecto.** | | | |
| ***Nombre*** | ***Empresa*** | ***Cargo*** | ***Fecha*** |
| Gerente General | GYM S.A. | Gerente General | 09-08-2021 |

**Diagrama de Gant**

Imagen que contiene Calendario

Descripción generada automáticamente

**Propuesta de valor**

****

**Modelo Lean Canvas**

