FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

“REDISEÑO E IMPLEMENTACIÓN DE LA CAPA DE PRESENTACIÓN DE UNA APLICACIÓN SOFTWARE DE VENTAS Y FACTURACIÓN PARA GARANTIZAR MAYOR USABILIDAD”

TESIS

PARA OPTAR EL TITULO DE INGENIERO DE SISTEMAS COMPUTACIONALES

Autor **:** Br. Victor Manuel Chavez Sanchez

Asesor **:** Ing. Lain Cárdenas Escalante

TRUJILLO – PERÚ

2018

DEDICATORIA

A nuestro Padre Celestial por darme la vida y la oportunidad de realizar mis metas.

A mis padres Manuel y Gladys, por darme la vida. A mis Hermanos Diego y Sandra por apoyarme a seguir adelante.

Agradezco a Dios por iluminarme con su luz divina durante todo este tiempo de mi vida y por brindarme la oportunidad de conseguir este logro.

A mi enamorada Riccy, por su apoyo incondicional y siempre darme las fuerzas necesarias para continuar.

A mi asesor, por el tiempo brindado para apoyarme en el desarrollo de la presente investigación.

RESUMEN

El presente trabajo tuvo como objetivo general, mejorar la usabilidad de un software mediante la propuesta de rediseño e implementación de la capa de Presentación de un software de Ventas y Facturación.

Para lo cual, en primera instancia se ha descrito la realidad actual de las empresas, este análisis nos ayuda a identificar que el uso de Software actualmente está en incremento cada año, las empresas buscan mejorar su productividad y por ende recurren a sistemas informáticos los cuales están en constante cambio y mejoramiento. También gracias a los estudios realizados por Jakob Nielsen, se estima que el esfuerzo necesario para aplicar técnicas y métodos de la ingeniería de usabilidad al proceso de desarrollo es de dos personas-año, y como mucho de cuatro personas-año.

El mapeo actual mostraba una mantenibilidad de proyecto de 64 puntos, una complejidad ciclomatica de 804 y a su vez 804 líneas de código. Por otro lado, luego de los cambios realizados en el software la mantenibilidad subió a 69 puntos, la complejidad se redujo a 729 puntos y las líneas de código se redujeron a 8646.

Adicionalmente utilizando la herramienta SonarQube nos da una nota aprobatoria en la evolución del software, además de la reducción de la duplicidad de código de 7.8% a 6.4%.

ABSTRACT

The general objective of this work was to improve the usability of software through the proposal of redesign and implementation of the Presentation layer of a Sales and Billing software.

In the first instance, the current reality of the companies has been described, this analysis helps us to identify that the use of Software is currently increasing every year, companies seek to improve their productivity and therefore resort to computer systems which are constantly changing and improving. Also thanks to the studies carried out by Jakob Nielsen, it is estimated that the effort necessary to apply techniques and methods of usability engineering to the development process is two people-years, and at most four people-years.

The current mapping showed a project maintainability of 64 points, a cyclomatic complexity of 804 and 804 lines of code. On the other hand, after the changes made in the software the maintainability aumented to 69 points, the complexity was reduced to 729 points and the lines of code were reduced to 8646.

Additionally using the tool SonarQube gives us a note of approval in the evolution of the software, in addition to the reduction of code duplication from 7.8% to 6.4%.

ÍNDICE DE CONTENIDOS

1. DATOS PRELIMINARES

1.1. Facultad

Facultad de Ingeniería.

1.2. Carrera profesional

Ingeniería de Sistemas Computacionales.

1.3. Título de la investigación

Rediseño e implementación de la capa de presentación de una aplicación software de ventas y facturación para garantizar mayor usabilidad.

1.4. Autor(es)

Br. Victor Manuel Chavez Sanchez

chavezsanchezv@gmail.com

1.5. Asesor

Ing. Lain Cárdenas Escalante

Magister en Ciencias Computacionales

laincardenas@gmail.com

1.6. Tipo de investigación

1.6.1. Según el propósito.

La Investigación es Aplicada

1.6.2. Según el diseño de investigación.

La Investigación es: Pre Experimental

1.7. Localización

1.7.1. Institución donde se desarrollará el proyecto.

- ***El trabajo de campo o aplicación:***

  Universidad Privada del Norte

- ***Las tareas de gabinete***

  Ingeniería de Sistemas Computacionales, en la UPN.

1.7.2. Distrito, Provincia, Región.

Trujillo, Trujillo, La Libertad

1.8. Alcance

La presente es una investigación Pre Experimental, en la cual se pretende rediseñar e implementar la capa de Presentación de un Software, esto facilitará su uso y manejo a los usuarios, además mejorará la productividad de estos debido al fácil entendimiento, mejor comprensión y la rápida recuperación ante errores cometidos por los usuarios en su uso.

1.9. Recursos

ITEM	Unidad	Cantidad
A. Humanos		
Asesor		1
Investigador		1
B. Material de oficina y escritorio		
Papel Bond (Tamaño A4)	Millar	1
Cuadernos	Unidad	1
Bolígrafos	Unidad	2
Tinta para Impresión color negro	Unidad	1
Tinta de impresión a colores	Unidad	3
C. Equipos		
Laptop	Unidad	1
PC Escritorio	Unidad	1
D. Servicios		
Internet	Horas	432
Fotocopias	Unidad	100
Movilidad (Pasajes)		50

Tabla 1 - Lista de Recursos

1.10. Presupuesto

ITEM	Unidad	Cantidad	Costo Unitario (S/.)	Costo Total (S/.)
A. Material de oficina y escritorio				
Papel Bond (Tamaño A4)	Millar	1	S/.21.00	S/.21.00
Cuadernos	Unidad	1	S/.5.00	S/.5.00
Bolígrafos	Unidad	2	S/.1.50	S/.3.00
Tinta para Impresión color negro	Unidad	1	S/.15.00	S/.15.00
Tinta de impresión a colores (magenta, azul y amarillo)	Unidad	3	S/.17.00	S/.51.00
Subtotal				S/.95.00
B. Otros servicios				
Internet	Horas	432	S/.1.00	S/.432.00
Fotocopias	Unidad	100	S/.0.10	S/.10.00
Movilidad (Pasajes)				S/.0.00
Subtotal				S/.442.00
Imprevistos				S/.0.00
TOTAL				*S/.537.00*

Tabla 2 - Presupuesto

1.11. Financiamiento

Financiamiento Propio

1.12. Cronograma

ETAPAS	FECHA		MESES				
	Inicio	Fin	Agosto	Setiembre	Octubre	Noviembre	Diciembre
Recolección de datos y revisión de la información	02/08/2017	02/10/2017	X	X	X		
Formulación y sustentación del proyecto y de investigación	05/09/2017	05/10/2017		X	X		
Diseño e implementación de la capa de presentación	10/10/2017	01/12/2017			X	X	X
Evaluación e interpretación de resultados	15/11/2017	01/12/2017				X	X
Redacción y corrección del informe y software	15/11/2017	01/12/2017				X	X
Presentación y Sustentación	10/12/2017	20/12/2017					X

Tabla 3 -Cronograma del proyecto

2. PLAN DE INVESTIGACIÓN

2.1. Problema de Investigación

2.1.1. Realidad Problemática

En una investigación realizada el año 2006 en España, Jakob Nielsen en sus estudios sobre 31 proyectos de desarrollo, estima que el esfuerzo necesario para aplicar técnicas y métodos de la ingeniería de usabilidad al proceso de desarrollo es de dos personas-año, y como mucho de cuatro personas-año. (Mario, 2006)

En muchas ocasiones es el usuario el que necesita adaptarse a la aplicación. Esto último es imperdonable en estos tiempos, en los que la competitividad en el mundo de las tecnologías de la información está a tan alto nivel.

Teniendo en cuenta los resultados preliminares de un estudio exploratorio realizado en PyMes de software de la Ciudad de Corrientes en Argentina, se llegó a la conclusión de que las empresas desconocen la importancia de la usabilidad, e incorporan algunas técnicas específicas (Maximiliano, Cristina & Raquel, 2013), esto dado que prestan mayor atención a elementos relacionados con el interior del sistema como su rendimiento o confiabilidad, por lo que aspectos tan relevantes como realizar un diseño interactivo centrado en mantener la atención del receptor, que se adapte a las características específicas de cada usuario, y que muestre rápidamente la información solicitada, han sido delegados a un segundo plano.

En la actualidad el uso de sistemas de información es cada vez más habitual en grandes y pequeñas empresas. Según un estudio realizado por el INEI se puede observar en el Gráfico 1.2 un incremento del 2% en el año 2014 frente a los resultados mostrados en el Grafico 1.2 del año 2013 en lo que respecta al uso de computadoras para el manejo de información.

PERÚ: USO DE COMPUTADORAS EN LAS EMPRESAS, SEGÚN SEGMENTO EMPRESARIAL

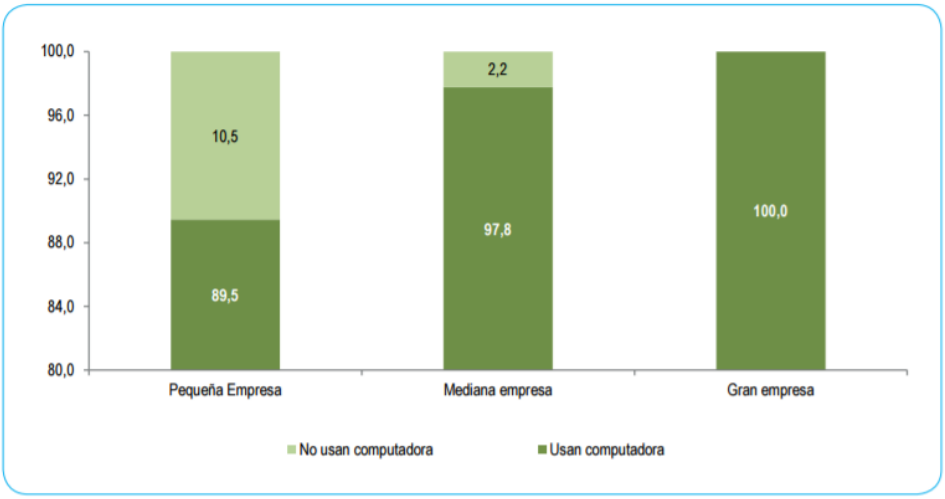


Grafico 1.1 Porcentaje de uso de computadores según segmento empresarial

Fuente: Instituto Nacional de Estadística e Informática - Encuesta Económica Anual 2013.

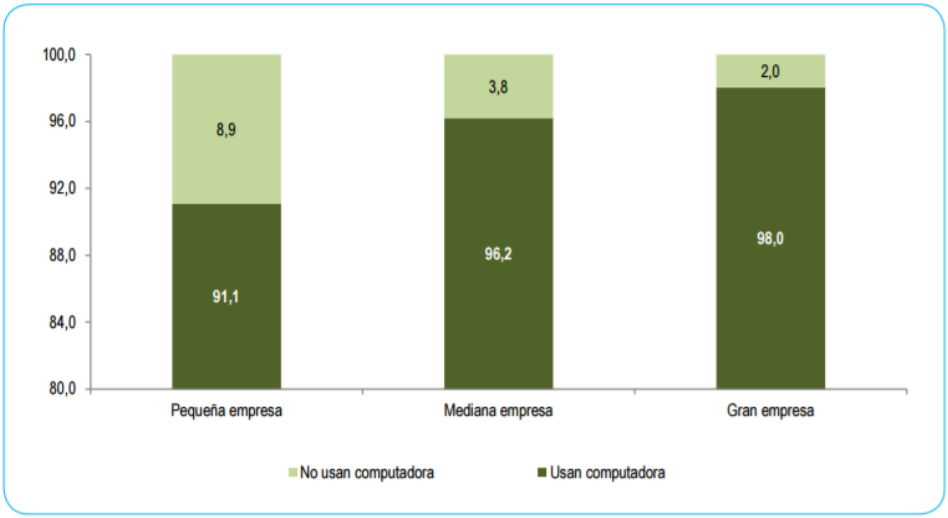


Grafico 1.2 Porcentaje de uso de computadores según segmento empresarial.

Fuente: Instituto Nacional de Estadística e Informática - Encuesta Económica Anual 2014.

Asimismo, en el mismo informe emitido por el INEI, se obtiene que las empresas están invirtiendo más en la Tecnología, a continuación, en el grafico 1.4 se observa un incremento de más del 2% en el caso de las Pequeñas Empresas y un incremento de 7% en el caso de las grandes empresas frente a los resultados mostrados por el grafico 1.2 correspondiente al año 2013.

PERÚ: EMPRESAS QUE REALIZARON INVERSIÓN EN CIENCIA Y TECNOLOGÍA, SEGÚN SEGMENTO EMPRESARIAL

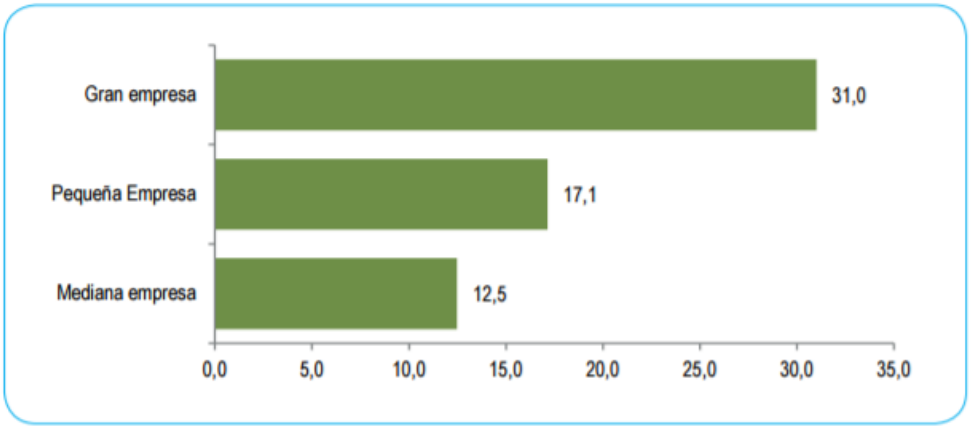


Grafico 1.3 Porcentaje de inversión en Tecnología

Fuente: Instituto Nacional de Estadística e Informática - Encuesta Económica Anual 2013.

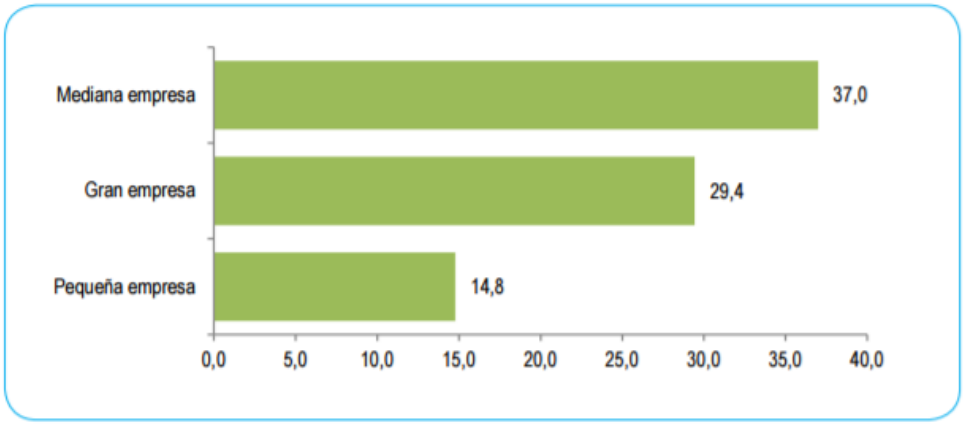


Grafico 1.4 Porcentaje de inversión en Tecnología

Fuente: Instituto Nacional de Estadística e Informática - Encuesta Económica Anual 2014.

Esta inversión que realizan las empresas conlleva a el desarrollo de sistemas de información, esto con el fin de controlar la gran cantidad de información que pueden manejar día a día entre las cuales podemos mencionar información contable, planillas, ventas y entre muchas más.

El éxito del desarrollo de un software depende de su usabilidad y utilidad para los usuarios, un software puede fracasar totalmente si dichos elementos no son sólidos al momento de la presentación de un sistema de información.

Aunque contamos con herramientas que nos ayudan a realizar un análisis de usabilidad en los sistemas informáticos, tales como: Usabilla, User Plus,

Noldus Observer, entre otras; muy pocas veces son aplicadas en el desarrollo de sistemas de información, y a su vez estas en su mayoría aplicadas a desarrollos web.

Uno de los problemas más comunes es el poco interés por parte de la desarrolladores de software para desarrollar sets de pruebas de usabilidad necesarias, ocasionando que el software sea poco intuitivo para el cliente, un claro ejemplo es la poca información que se puede obtener de un mensaje emitido por el sistema por una mala acción del usuario al realizar un proceso especifico, por ejemplo ingresar un símbolo en vez del número de DNI, dificultando así que el usuario puede identificar como solucionar el problema.

Asimismo, ocurre con la cantidad de data mostrada en las interfaces, ya que, un claro ejemplo es el uso de teclas abreviadas que según el grafico 1.5 podemos observar que no está presente, que muchas veces los sistemas no las tienen o en su defecto las tienen, pero no respetando el estándar, ejemplo: control+ i en vez de control+p (imprimir).

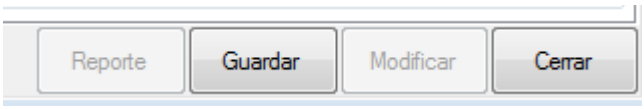


Grafico 1.5 Teclas sin Abreviaturas dificultan el uso del Software

Fuente: Evaluación a Sistema realizada, no se observan teclas abreviadas.

Según un primer análisis del software a mejorar, se pudo detectar problemas en la forma como se ingresa el texto al sistema, entre los cuales podemos citar los formularios de registro de un cliente, articulo, proveedor, etc. Como ejemplo el sistema permite ingresar “X” caracteres en un DNI/RUC o letras en un número telefónico, esto puede generar confusión en el usuario y asimismo pérdida de productividad al verse necesario corregir el error.

También se debe considerar no mostrar información que no es de importancia para el usuario en los mensajes de error, hay casos en que no se está aplicando lo anteriormente comentado en el sistema a mejorar, los

mensajes de error no son muy amigables para los usuarios no avanzados, pudiendo generar así que el usuario no se sienta cómodo con el sistema.

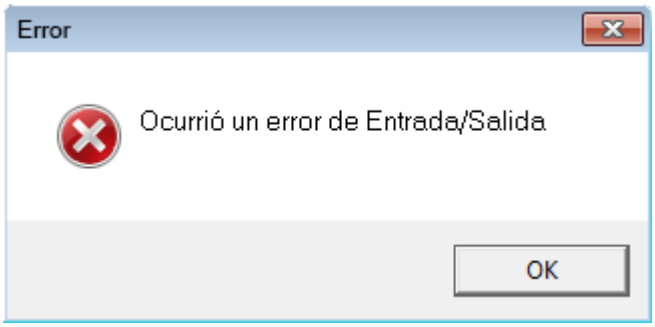


Grafico 1.6 Mensaje emitido por le sistema a rediseñar.

Fuente: Mensaje obtenido realizando las pruebas en el sistema a rediseñar.

Todas las evaluaciones realizadas al sistema están presentes en la sección Apéndice del presente informe, así como los resultados de las pruebas.

La capa de presentación al ser la que interactúa directamente con el usuario final, puede mejorar considerablemente la realización de un proceso específico, así como también disminuir los posibles errores que se pudieran cometer en el transcurso de un proceso.

2.1.2. Formulación del problema

¿El rediseño e implementación de la capa de presentación permitirá garantizar una mayor usabilidad de una aplicación software de Ventas y Facturación?

2.1.3. Justificación del problema

- ***Aplicativa***

  El presente proyecto pretende mejorar el desarrollo de los principales procesos de la empresa a través del rediseño de la Capa de presentación, partiendo de un análisis exhaustivo en el uso del sistema que permita dar un diagnóstico sobre la situación actual de la capa de Presentación, para luego tomar las acciones pertinentes en cuanto al nuevo diseño.

- ***Valorativa***

  El uso de herramientas de software para el rediseño del presente software, permitirá evitar futuros incidentes en el uso de las funcionalidades del software y así el ahorro en gastos ante una posible reestructuración del software o caso contrario el precio que costaría reemplazarlo por uno nuevo.

- ***Académica***

  El desarrollo de este proyecto de tesis permitirá mejorar la productividad y el manejo de un software de Ventas, gracias a que se rediseñara la capa de presentación del software utilizando algunas herramientas tecnológicas (C#, asp.net) la cual permitirá a los usuarios culminar de manera exitosa y sin demoras el uso de alguna funcionalidad en el software.

2.1.4. Limitaciones

Se cuenta con un corto periodo de tiempo, que es 1 año, para Rediseñar e implementar la capa de presentación de un software, por lo tanto, se presentará un prototipo el cual cuente con las principales modificaciones realizas en la capa de Presentación y de las principales funcionalidades del sistema.

2.1.5. Objetivos

- ***Objetivo General***

  Rediseñar e implementar la capa de presentación de una aplicación software de ventas y facturación para garantizar mayor usabilidad.

- ***Objetivos Específicos***

  -Identificar los problemas que dificultan la labor de los usuarios en la Capa de Presentación

  -Rediseñar la interfaz de usuario para disminuir el tiempo de aprendizaje del uso de las funciones.

  -Rediseñar la interfaz de usuario para garantizar una mayor comprensibilidad al utilizar la aplicación software de ventas.

  -Refactorizar la lógica de presentación para asegurar una mayor mantenibilidad del software.

2.2. Marco Teórico

2.2.1. Antecedentes

El estudio realizado por los estudiantes Eckbert Schulz, Laybet Colmenares, Mariela Salzano, Johaira Romero, Diego Centeno, Hyxia Villegas (2008) en la Universidad de Castilla de La Mancha titulada “Metodología para la Evaluación de la Usabilidad y Mejora de la Interfaz del Material Educativo”, se plantea el desarrollo de una metodología para la evaluación de la usabilidad y mejora de la interfaz del material educativo computarizado (MEC) consiste de un ciclo iterativo de evaluación, mejora y desarrollo de prototipo, donde se evalúa la usabilidad del programa mediante la inspección realizada por expertos siguiendo la Heurística de Nielsen, y las evaluaciones empíricas: Observación de Campo y Test Piense en Voz Alta; el resultado de estos ciclos de evaluación fue la detección de alrededor de 85% de los errores y la producción de tres prototipos de la interfaz, incluyendo al prototipo final, el cual tuvo una alta aceptación por parte de los usuarios. En la investigación se lograron tres prototipos de la interfaz, que evidencian la evolución hasta llegar al prototipo final, el cual tuvo una alta aceptación por parte de los usuarios. Al aplicar la metodología cíclica de

evaluación y mejora, centrada en el usuario, se logró generar una interfaz que cumple con los principios de usabilidad, ya que su diseño se apegó a los principios de Nielsen y proveen una alta usabilidad de la interfaz, la cual permite realizar el trabajo de forma eficiente, fácil de aprender y de recordar, permite, además, prevenir errores y ofrece alta satisfacción en el aprendizaje.

Esta investigación nos muestra la importancia de la usabilidad en las aplicaciones software, en especial el impacto que puede tener en nuestra capa de Presentación, específicamente la interfaz de usuario. Asimismo, se utilizó la filosofía Diseño centrado en el usuario, que se basa en las capacidades del usuario para el manejo del software, dicha filosofía está ligada con la Capa de Presentación.

Teniendo en cuenta otros estudios como el realizado por Mario Lorenzo Alcala (2006) en la Universidad Nacional de Educación a Distancia de España titulada “Medida de la usabilidad en aplicaciones de Escritorio. Un método práctico”, Mario Lorenzo nos muestra que la interfaz de usuario es una parte importantísima en el desarrollo de aplicaciones informáticas y es clave en el éxito o fracaso a la hora de la puesta en producción de la aplicación en el mundo real. A través de ella, el usuario interactúa con la aplicación con más o menos dificultad. Esta mayor o menor dificultad es determinante cuando una empresa elige sus aplicaciones empresariales porque de ellas dependerá de forma muy importante su productividad y costes.

A la vista de los resultados obtenidos en dicho informe como consecuencia de la aplicación de la métrica, se puede concluir que la métrica propuesta es válida para su aplicación en el desarrollo de sistemas interactivos y cuantificar la usabilidad de estos sistemas de una forma más económica que las típicas evaluaciones heurísticas donde se emplea a diferentes evaluadores y usuarios.

En dicho trabajo se plantean heurísticas y métricas de usabilidad muy interesantes, que si bien es cierto fueron aplicadas software de grandes

empresas tales como IBM podrían ser aplicadas a software para pequeña empresas.

Asimismo en función de los resultados se puede deducir que las grandes compañías producen mejores aplicaciones informáticas como consecuencia, seguramente, de la aplicación de más recursos a su desarrollo, que por otra parte resulta bastante lógico.

Por último podemos mencionar la investigación realizada por Juan Angel Martínez Párraga (1999) de la Universidad de Castilla de España titulado "Estándar IEEE 1219 de Mantenimiento del Software". En el estudio sobre el Estándar IEEE 1219 realizado por Juan Martínez en el año 1999, se detalla cada una de las definiciones del Estándar, a través de ejemplos claros y sencillos. Nos explica acerca de la planificación, fases, gestión y características del mantenimiento de software. El trabajo nos ayudara a entender la IEEE 1219 y poder escoger el mantenimiento más adecuado para aplicarlo en nuestro rediseño y/o implementación de la Capa de Presentación del software a Evaluar.

2.2.2. Bases Teóricas

a) Capa de Presentación:

- ***Definición:***

  Es la responsable de la presentación visual de la aplicación. La capa de presentación enviará mensajes a los objetos de esta capa de negocios o intermedia, la cual o bien responderá entonces directamente o mantendrá un diálogo con la capa de la base de datos, la cual proporcionará los datos que se mandarían como respuesta a la capa de presentación. (Roger Pressman, 2010)

  Podemos decir que es la única capa a la que el usuario tiene acceso, el usuario debe ser capaz de asimilar la información asimilada rápidamente y de manera sencilla.

Tiene como función básica encargarse del formato en que se va a mostrar la información. Establece el contexto entre elementos del nivel de aplicación, determinando la sintaxis y la semántica de los símbolos empleados para representar la información, traduciendo el formato de aplicación al formato de red y viceversa.

Esta definición nos ayudara a entender el propósito y la función de la capa de presentación, permitiéndonos así centrarnos en los puntos más importantes para poder ser aplicadas en el software al cual se realizara el rediseño en su capa de Presentación.

- ***Características:***
  - Permite formatear datos y mostrarlos en estilos visuales útiles.
  - Navegar, buscar y organizar los datos mostrados.
  - Permite administrar diseños visuales, estilos y la apariencia general y navegación de la aplicación.
  - Permite la Validación, enmascaramiento de entrada y uso de controles apropiados para la entrada de datos.
- ***Componentes:***
  1. *Componentes interfaz de usuario:* Estos son los elementos visuales de la aplicación utilizados para mostrar información al usuario y aceptar la entrada del usuario.
  2. *Componentes de lógica de presentación:* La lógica de presentación es el código de la aplicación que define el comportamiento lógico y la estructura de la aplicación de una manera que es independiente de cualquier implementación de interfaz de usuario específica.
- ***Patrones para el Diseño de la Capa de Presentación:***
  1. *Escoger la Tecnología apropiada:* Para el caso de Windows solo tenemos 2 opciones: Windows Forms and WPF.
  2. *Diseño de IU y Estructura:* El diseño de una aplicación puede significar la diferencia entre una aplicación que es fácil de usar y eficiente y una aplicación que es complicada y frustra al usuario.

La cuidadosa evaluación del diseño de la aplicación es vital para atender las necesidades de sus usuarios.

En el libro *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design,* authors Larry Constantine and Lucy Lockwood se describen los siguientes principios para el diseño de UI:

- **Diseño de la Estructura:** Una interfaz de usuario bien estructurada es lógico, coherente e intuitiva. Se debe utilizar un modelo en el diseño de la interfaz de usuario. Poner los elementos con funcionalidad relacionada juntos en grupos organizados lógicamente. Los grupos funcionales deben ser fácilmente reconocible por el usuario, y los artículos diferentes o grupos funcionales deben ser diferenciados fácilmente. Una interfaz de usuario bien estructurado no tendrá usuarios buscando a ciegas una la funcionalidad deseada, pero les permitirá encontrar lo que necesitan de manera intuitiva.

- **Diseño para simplicidad:** Una interfaz de usuario bien diseñada es simple y fácil de usar. Las tareas comunes son de fácil acceso y la navegación para las tareas utilizadas con poca frecuencia es más sencillo e intuitivo. El uso de elementos de menú para las tareas comunes es un ejemplo de la incorporación simplicidad en una interfaz de usuario.

- **Diseño para visibilidad:** Una buena interfaz de usuario tendrá unidades funcionales necesarias fácilmente visibles y accesibles para el usuario. Es bueno mantener la simplicidad en mente con este principio.

- **Diseño para retroalimentación:** Una buena UI deberá mantener informado al usuario respecto a los cambios en el estado de la aplicación. Los errores que son relevantes para el usuario o acciones que deban llevarse a cabo por el usuario se deben comunicarse con claridad y concisión. Por el contrario,

si no se requiere acción alguna del usuario, no debe ser innecesariamente expuesta.

- **Diseño por tolerancia:** Los usuarios cometen errores. Una interfaz de usuario bien diseñada permitirá al usuario recuperarse rápida y fácilmente de los errores. Las interfaces de usuario deben permitir que la entrada defectuosa deba ser fácil y rápidamente corregida.

- **Diseño para reusar:** Una interfaz de usuario debe ser coherente. Los componentes y estilos visuales deben ser reutilizados siempre que sea posible. Esto no sólo reduce la cantidad de código.

3. *Diseñar Flujo de Trabajo de la aplicación:* Algunas aplicaciones no requieren un flujo de trabajo dirigido o tienen requisitos de flujo de trabajo muy simplistas.

   - **Implementar navegación de usuario:** Interfaces de usuario simples pueden que no requieran más de un solo formulario, pero para las interfaces más complejas una interfaz de navegable de usuario puede ser requerida.

   - **Uso de PageFunction:** La clase PageFunction es muy similar a la clase Page. Se puede diseñar un PageFunction en el diseñador, se puede añadir controles a un PageFunction, y se puede navegar en una PageFunction a través de hiperenlaces o utilizando NavigationService.

4. *Diseño de la entrada y presentación de la información:* La mayor parte de lo la capa de presentación está haciendo en una aplicación es bien mostrar o recibir datos. La capa de presentación es la responsable de la validación de los datos, el enlace de datos y su presentación al usuario.

5. *Diseño de la capacidad de respuesta del UI*: Mientras que la función principal de la capa de presentación es interactuar con el usuario, también se puede utilizar la potencia de la computadora del cliente para el manejo de las tareas informáticas que no requieren la comunicación con las capas de servidor o de datos. Utilizando técnicas multihilo se permite utilizar la potencia de procesamiento del cliente, manteniendo la capacidad de respuesta de la interfaz de usuario.

b) Usabilidad

- ***Definición:***

  El término “usabilidad” deriva del inglés “Usability”, es un atributo cualitativo definido comúnmente como la facilidad de uso, ya sea de una página Web, una aplicación informática o cualquier otro sistema que interactúe con un usuario. El concepto generalmente se refiere a una aplicación informática o un aparato, aunque también puede aplicarse a cualquier sistema hecho con algún objetivo particular. También se refiere a métodos para mejorar la facilidad de uso durante el proceso de diseño. *(Walter Sanchez, 2011):*

  Jakob Nielsen (2012), considerado el padre de la usabilidad, la definió como “el atributo de calidad que mide lo fáciles de usar que son las interfaces Web”. Es decir un sitio Web usable es aquél en el que los usuarios pueden interactuar de la forma más fácil, cómoda, segura e inteligente posible. Como se dijo antes, en la usabilidad se debe cumplir con algunas características que logren que el usuario encuentre lo que busca en el menor tiempo posible y por ende, el contenido y la estética deben ser el principal foco. Si bien la visibilidad también afecta la usabilidad, algunos atributos como entendible, novedoso, comprensible, inteligente y atractivo harán los contenidos más cercanos al usuario, ayudarán en la navegación intuitivamente y por ende, su experiencia al enfrentarse a la pantalla debería ser mucho más placentera, evitando en lo posible los clicks y scroll.

- ***Características***

  - Facilidad de aprendizaje: define en cuánto tiempo un usuario, que nunca ha visto una interfaz, puede aprender a usarla bien y realizar operaciones básicas.
  - Facilidad y Eficiencia de uso: determina la rapidez con que se pueden desarrollar las tareas, una vez que se ha aprendido a usar el sistema.

- Facilidad de recordar cómo funciona: se refiere a la capacidad de recordar las características y forma de uso de un sistema para volver a utilizarlo a futuro.
- Frecuencia y gravedad de errores: plantea la ayuda que se le entrega a los usuarios para apoyarlos cuando deban enfrentar los errores que cometen al usar el sistema.
- Satisfacción subjetiva: indica lo satisfechos que quedan los usuarios cuando han empleado el sistema, gracias a la facilidad y simplicidad de uso de sus pantallas.

- ***Componentes:***

Según Jakob Nielsen la usabilidad está divido en 5 componentes de calidad, las cuales son (Jakob Nielsen, 2012):

- Aprendibilidad

  Examina lo fácil que es para un usuario realizar tareas básicas la primera vez que interactúan con su producto. ¿Luchan con una pieza en particular? ¿El usuario se frustra con la forma de llevar a cabo esta tarea? ¿Afecta el resto de la experiencia?

- Eficiencia

  Analiza la capacidad de los usuarios para realizar rápidamente una tarea una vez que ya conocen el diseño. ¿Demoran demasiado para ejecutar tareas sencillas?

- Memorabilidad

  Si un usuario ha pasado algún tiempo lejos de su diseño y luego vuelve a interactuar con él, qué tan rápido puede restablecer la competencia. Si el diseño es intuitivo entonces el tiempo de reaprendizaje debe ser pequeño.

- Errores

  El seguimiento de cuántos errores hacen los usuarios, la gravedad de estos errores y la facilidad con que pueden recuperarse de estos, puede ayudarle a determinar los puntos de dolor dentro de su diseño.

- Satisfacción

  Determinar qué tan agradable es el diseño, este es el componente capaz de determinar si los usuarios vuelven a su producto, o no.

Cuando tengamos las respuestas a estos cinco componentes, podremos responder las preguntas adicionales. Por ejemplo, ¿qué características necesita realmente poner en su producto? ¿Qué tan fácil y agradable son estas características para usar? Finalmente, ¿son útiles estas características?

- ***Métodos de Evaluación:***

  Actualmente existe una variedad de métodos utilizados para evaluar la usabilidad. Algunos de estos métodos hacen uso de datos recolectados de usuarios y sus preferencias, mientras que otros confían en los expertos en usabilidad.

  Cuando se decide seleccionar un método, se debe considerar el costo, las restricciones de tiempo y la idoneidad del método. (Walter Sanchez, 2011)

  Los métodos de la usabilidad, pueden ser clasificados en las siguientes subcategorías.

  - **Métodos de modelado cognitivos:**

    Los modelos cognitivos consisten en crear un modelo computacional para estimar cuánto tiempo le toma a la gente realizar una determinada tarea. Los modelos están basados en principios psicológicos y estudios experimentales para determinar los tiempos de procesamiento cognitivo y movimientos motores. Los modelos cognitivos pueden utilizarse para mejorar interfaces de usuario o predecir errores y dificultades durante el proceso de diseño.

    Algunos modelos:

    a. **Diseño Paralelo:** Con éste, varias personas crean un diseño inicial del mismo conjunto de requerimientos. Cada persona trabaja en forma independiente, y cuando termina, comparte sus conceptos con el grupo. El equipo de diseño

considera cada solución, y cada diseñador utiliza las mejores ideas para mejorar aún más su propia solución.

b. **GOMS:** Es un acrónimo que significa objetivos (Goals), operador (Operator), métodos (Methods) y reglas de selección (Selection Rules). Se trata de una familia de técnicas que analiza la complejidad de los sistemas interactivos de usuarios. Los objetivos (Goals) son lo que el usuario tiene que llevar a término.
c. **Modelo de Procesador Humano:** A veces es útil desglosar una tarea y analizar cada aspecto separadamente. Esto permite a quien está haciendo las pruebas, localizar áreas específicas de mejora. Para ello, es necesario comprender cómo el cerebro humano procesa la información.

- **Métodos de inspección:** Se basa en la disponibilidad de evaluadores que examinan si una interfaz determinada cumple una serie de principios de usabilidad. Estos métodos dependen de las opiniones, juicios e informes generados por los evaluadores de usabilidad.

- **Métodos de investigación:** Los siguientes métodos de evaluación de usabilidad implican la recopilación de datos cualitativos de los usuarios. Aunque los datos recogidos son subjetivos, proporcionan valiosa información sobre lo que el usuario desea. (Walter Sanchez, 2011)

a. **Análisis de Tareas:** Con éste, varias personas crean un diseño inicial del mismo conjunto
b. **Grupos de Enfoque:** Es un acrónimo que significa objetivos (Goals), operador (Operator), métodos (Methods) y reglas de selección (Selection Rules). Se trata de una familia de técnicas que analiza la complejidad de los sistemas interactivos de usuarios. Los objetivos (Goals) son lo que el usuario tiene que llevar a término.
c. **Modelo de Procesador Humano:** A veces es útil desglosar una tarea y analizar cada aspecto separadamente. Esto

permite a quien está haciendo las pruebas, localizar áreas específicas de mejora. Para ello, es necesario comprender cómo el cerebro humano procesa la información.

- **Métodos de Prototipito:**

  a. **Prototipado Rápido:** El prototipo rápido se describe como un método basado en ordenador que pretende reducir el ciclo iterativo de desarrollo. Los prototipos iterativos desarrollados podrán ser rápidamente reemplazados o modificados según los informes de diversas procedencias, como experiencias previas de usuarios o de diseñadores veteranos, a medida que se evoluciona en el desarrollo de las tareas a realizar. (Alejandro Floría, 2000)

- **Métodos de Ensayo:** Estos métodos de evaluación de usabilidad involucran prueba de temas para la mayoría de datos cuantitativos. Por lo general proporcionan tiempo de finalización de tareas y permite la observación de actitudes.

- ***Refactorización:***

  Es una técnica disciplinada para reestructurar las líneas de código, alterando su estructura interna sin cambiar su comportamiento externo. (Martin Fowler, 2013)

  La refactorización es la parte del mantenimiento del código que no arregla errores ni añade funcionalidad. Su objetivo es mejorar la facilidad de comprensión del código y diseño, y eliminar código muerto, para facilitar el mantenimiento en el futuro. Añadir nuevo comportamiento a un programa puede ser difícil con la estructura dada del software, así que un desarrollador puede refactorizarlo primero para facilitar esta tarea.

  Aunque la limpieza de código se lleva realizando desde hace décadas, el factor clave de la refactorización es realizar de manera intencionada la limpieza separándola de la adición de funcionalidad nueva, usando un catálogo conocido de métodos útiles de refactorización, para después comprobar el código ejecutando las pruebas unitarias, sabiendo que

cualquier cambio en el comportamiento significa que se ha introducido un error

- **Tipos de Refactorizacion:**

  Algunos tipos de refactorización detallados por Martin Fowler son:

  - Consolidación de Fragmentos de Condicionales Duplicados
  - Descomposición de condicionales
  - Extracción de Métodos
  - Renombre de los Métodos
  - Eliminación de Parámetros
  - Reemplazo de Códigos de Error con Excepciones
  - Extracción de clases
  - Extracción de variables
  - Encapsulamiento de Colectores
  - Extracción de Interfaces

Para este proyecto usaremos solo las que se adecuen a la codificación del proyecto.

2.2.3. Definición de términos básicos

Usabilidad: Cualidad de la página web o del programa informático que los hace sencillos de usar porque facilitan la lectura de los textos, descargan rápidamente la información y presentan funciones y menús sencillos, por lo que el usuario encuentra satisfechas sus consultas y cómodo su uso.

Capa de Presentación: la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario.

Interfaz de usuario: La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo. Normalmente suelen ser fáciles de entender y fáciles de accionar.

Componente de lógica de presentación: Actividades que crean un formulario en una aplicación. Entre ellas se incluyen el procesamiento de una solicitud, la generación de contenido de respuesta.

Comprensibilidad: La capacidad del producto software que permite al usuario entender si el software es adecuado y como puede ser usado para unas tareas o condiciones de uso particular.

- Funciones evidentes: que proporción de las funciones en el sistema son evidentes al usuario.
- Total de funciones entendidas: cantidad de funciones entendidas vs la cantidad de funciones
- Entendimiento de una función

Aprendizaje: La forma como el software permite al usuario aprender su uso. También es importante considerar la documentación.

- Facilidad de uso
- Facilidad de uso del manual de ayuda.
- Efectividad del manual de ayuda.

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

- Tiempo para encontrar información en una consulta.
- Tiempo de errores corregidos.
- Capacidad para corregir / revertir una función.

Visibilidad del estado del sistema. La aplicación debe mantener siempre informado al usuario del estado del sistema así como de los caminos que este pueda tomar con una retroalimentación visual apropiada en un tiempo

razonable. El sistema ofrecerá al usuario una respuesta que le indique lo que está sucediendo en cada una de las operaciones que realiza.

Consistencia y estándares. Una buena interfaz contribuye al aumento de la productividad si es consistente en todos los diálogos que desarrolla, basándose en el conocimiento que el usuario ha adquirido con otras aplicaciones y en la aplicación propia. Se debe mantener la consistencia en todas las aplicaciones relacionadas.

Prevención de errores. El mejor tratamiento de los errores es prevenirlos con un buen diseño de los diálogos desde el primer momento en que ocurren, minimizando los riesgos de que puedan ocurrir. Se debe realizar un buen diseño de mensajes de error que den la posibilidad al usuario de retraerse antes de que se realice la acción y se comprometan los datos.

Refactorización. Es una técnica disciplinada para reestructurar las líneas existentes de código, alterando su estructura interna sin cambiar su comportamiento externo, con el fin de hacer más fácil su entendimiento.

2.3. Hipótesis

2.3.1. Planteamiento de la hipótesis

El rediseño e implementación de la capa de presentación sí permite garantizar una mayor usabilidad de una aplicación software de Ventas y Facturación.

2.3.2. Variables

Variable Dependiente:

Usabilidad.

Variable Independiente:

Capa de presentación de una aplicación software de ventas y facturación.

2.3.3. Operacionalización de variables

VARIABLE	DEFINICIÓN CONCEPTUAL	DIMENSIONES	INDICADORES
Capa de presentación de una aplicación software de ventas y facturación.	La capa de presentación contiene los componentes que implementan y muestran la interfaz de usuario y administran la interacción del usuario. Esta capa incluye controles para la entrada y visualización del usuario, además de componentes que organizan la interacción del usuario. (Microsoft Patterns & Practices Team, 2009)	Componente de interfaz de usuario (vista)	-Incompatibilidad en la resolución de pantallas. -% de reutilización de controles en la interfaz. -Nivel de Organización de los elementos del formulario.
		Componente de Lógica de presentación. (Controlador)	-% de Reutilización de Código -Eliminación de código innecesario. -Validación de la información INPUT.
Usabilidad	La Usabilidad es la medida de la calidad de la experiencia que tiene un usuario cuando interactúa con un producto o sistema. Esto se mide a través del estudio de la relación que se produce entre las herramientas y quienes las utilizan, para determinar la eficiencia en el uso de los diferentes elementos ofrecidos en las pantallas y la efectividad en el cumplimiento de las tareas que se pueden llevar a cabo a través de ellas. (Unidad de Modernización y Gobierno Digital , 2008)	Comprensibilidad	- Total de las funciones en el sistema que son evidentes al usuario. -Cantidad de funciones entendidas vs la cantidad de funciones disponibles -Capacidad para diferenciar información de entrada y de salida en el sistema.
		Aprendizaje	-Tiempo de aprendizaje de una función específica.
		Operatividad	-Tiempo para corregir / revertir una función. -Número de funciones en el sistema que permiten ser canceladas.

Tabla 4 - Cuadro detallado de la Operacionalización de las Variables

2.4. Materiales y métodos

2.4.1. Tipo de diseño de investigación

Pre experimental.

2.4.2. Material de estudio

Para este proyecto la Unidad de Estudio es sólo el Software que se va a mejorar, por lo tanto, no hay población.

2.4.3. Técnicas, procedimientos e instrumentos

- ***De recolección de información***

Como primera instancia utilizaremos los Casos de Prueba en el software para saber si las características presentadas en el software con satisfactorias.

Se hará uso de todas las funciones disponibles en el sistema, para así conocer el propósito de esta y obtener la mayor cantidad de información disponible para aplicarla a las formulas o métricas. Este dato nos ayudara a evaluar la cantidad de funciones que fueron fácil de reconocer al usar por primera vez el software.

Se recolectará información de los datos de entrada y salida en el sistema, para esto ya se debe haber hecho uso de todas las funciones disponibles en el sistema.

Se probará todos y cada una de las validaciones disponibles, y los mensajes recibidos por parte del sistema, con el fin de descartar si los mensajes son entendibles para un usuario común sin necesidad de conocimientos avanzados. Se tomará nota de las validaciones erróneas o no disponibles en el software.

Así mismo ante cometer un error de validación, los mensajes que nos mostrara el sistema (en caso lo haga) deberían permitir solucionar el inconveniente.

Posteriormente usaremos la herramienta para procesar las métricas de código de la plataforma VisualStudio 2015 para analizar el código utilizado en la Capa de Presentación, con eso obtendremos las líneas de código, el índice de mantenimiento, la complejidad ciclomatica, profundidad de herencia y el acoplamiento de las clases.

Por últimos usaremos la plataforma SonarQube para entrar en el análisis manera más minuciosa de las líneas de código en la aplicación, asimismo nos brinda la opción de una antes y después de los cambios que nos puede ayudar a ver el % de código reducido, como dato adicional la plataforma también nos analiza el proyecto en su totalidad dándole una calificación.

- ***De procesamiento de información***

  Se aplicarán las métricas de calidad de software relacionas a la Usabilidad, planteadas en la ISO 9126.

  - Funciones del sistema Evidentes:

    $$X = A / B$$

    Donde, A = Número de funciones identificadas y B = Número de funciones disponibles en el sistema.

  - Funciones del sistema Entendidas:

    $$X = A / B$$

    Donde, A = Número de funciones que fueron entendidas por el usuario y B = Número de funciones disponibles en el sistema.

  - Entradas y salidas de datos entendidas:

$$X = A / B$$

Donde, A = Número datos de entrada y salida identificados y B = total de datos de entrada y salida.

- Efectividad del manual de usuario:

$$X = A / B$$

Donde, A = Tareas completadas con éxito después de usar el manual de ayuda y B = Total de funciones probadas.

- Correcciones/modificaciones de datos de entradas en los formularios.

$$T = A / B$$

Donde, A = Nro. De formularios donde se hizo modificaciones en los datos de entrada antes de ser culminada y B = Nro. De formularios en donde el usuario trato de hacer modificaciones durante el tiempo de respuesta de este.

- Proporción de funciones que son personalizables por el usuario:

$$X = A / B$$

Donde, A = Funciones personalizables de forma exitosa y B = Intentos de personalización de funciones.

- Tiempo de aprendizaje de una función:

T = Tiempo de aprendizaje.

- Funciones en el sistema que permiten ser canceladas por el usuario:

  $$X = A / B$$

  Donde, A = Funciones que pueden ser canceladas por el usuario, B = Total de funciones disponibles en el sistema.

- ***Instrumentos de recolección de datos:***

  - Observación
  - Diagrama de flujos
  - Diccionario de datos

3. DESARROLLO

El desarrollo del capítulo abarca la presentación del rediseño de la capa de presentación, detallando las actividades que se llevaron a cabo y las adecuaciones que se hicieron de acuerdo a la naturaleza de este proyecto.

3.1. Casos de Prueba - Usabilidad

Dado que en el sistema a rediseñar la capa de presentación no posee documentación, se procederá a realizar los Casos de Prueba de las principales funciones encontradas en el sistema.

3.1.1. Comprensibilidad

Como primer paso debemos identificar las funciones disponibles en el software. Para eso aplicaremos las métricas de la ISO 9126.

Como Menú Principal observamos lo siguiente:

Gráfico 3.1 Menú Principal de la Aplicación - Gestión

Con un primer vistazo podemos encontrar las siguientes funciones:

- Gestión:
  - Clientes
  - Personal
  - Proveedor
  - Artículos

Siguiendo con la secuencia, damos clic en la pestaña Comprobantes.

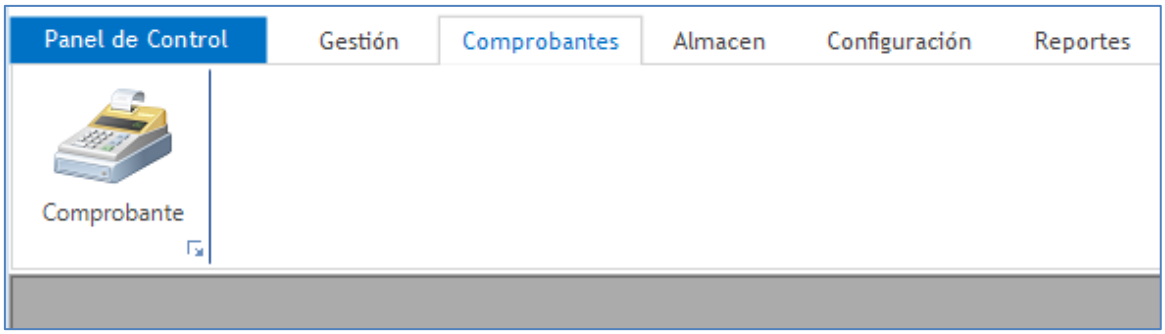


Gráfico 3.2 Menú Principal de la Aplicación - Comprobantes

En la que únicamente encontramos lo siguiente:

- Comprobantes:
  - Comprobante

A continuación pasamos a la pestaña Almacén.

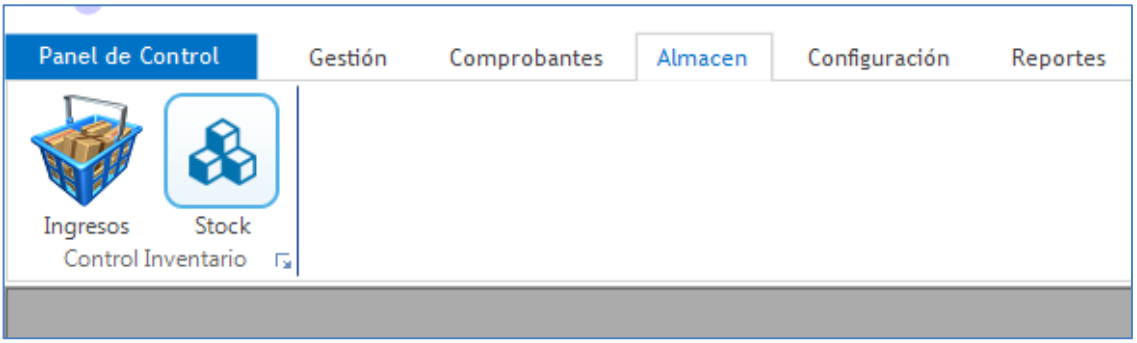


Gráfico 3.3 Menú Principal de la Aplicación - Almacén

- Almacén:
  - Ingresos
  - Stock

Continuamos con la pestaña Configuración.

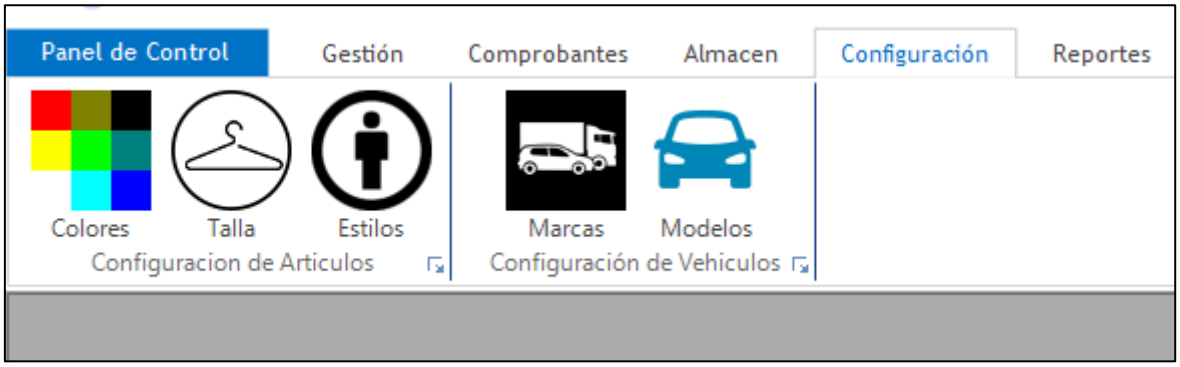


Gráfico 3.4 Menú Principal de la Aplicación - Configuración

- Configuración:
  - Colores
  - Talla
  - Estilos
  - Marcas
  - Modelos

Y por último tenemos la pestaña Reportes.

Gráfico 3.5 Menú Principal de la Aplicación - Reportes

- Reportes:
  - Ventas
  - Almacén
  - Tienda

Aplicando nuestros indicadores para la comprensibilidad obtenemos lo siguiente:

✓ **Funciones del sistema Evidentes:**

$$X = A / B$$

Donde, A = Número de funciones identificadas y B = Número de funciones disponibles en el sistema.

En el primer uso de la aplicación, logramos identificar un total de 15 funciones, de un total de 15.

$$X=15/15$$

Por lo tanto, el 100% de las funciones disponibles en el sistema fueron identificadas.

- ✓ **Funciones del sistema Entendidas:**

De las funciones identificadas inicialmente, definiremos la característica de cada uno.

- Gestión:
  - Clientes: Registro de Clientes
  - Personal: Registro de Personal Trabajador
  - Proveedor. Registro de Proveedores

- Reportes:
  - Ventas: Informe de ventas
  - Almacén: Stock en el almacén
  - Tienda: Stock en la tienda.

- Configuración:
  - Colores: Registro de colores
  - Talla: Registro de Talla
  - Estilos: Registro de Estilos
  - Marcas: Registro de Marcas
  - Modelos: Registro de Modelos

- Almacén:
  - Ingresos: Ingreso a Almacén
  - Stock: Productos disponibles

Por lo tanto:

$$X = A / B$$

Donde, A = Número de funciones que fueron entendidas por el usuario y B = Número de funciones disponibles en el sistema.

$$X = 15/ 15$$

Por lo tanto, el 100% de las funciones disponibles en el sistema fueron entendidas por el título mostrado en el menú principal.

✓ **Entradas y salidas de datos entendidas:**

A continuación, estudiaremos cada pantalla en específico según cada función disponible en el sistema, esto nos ayudara a comprobar si la organización de los formularios es la más adecuada.

- Gestión:
  - Clientes:

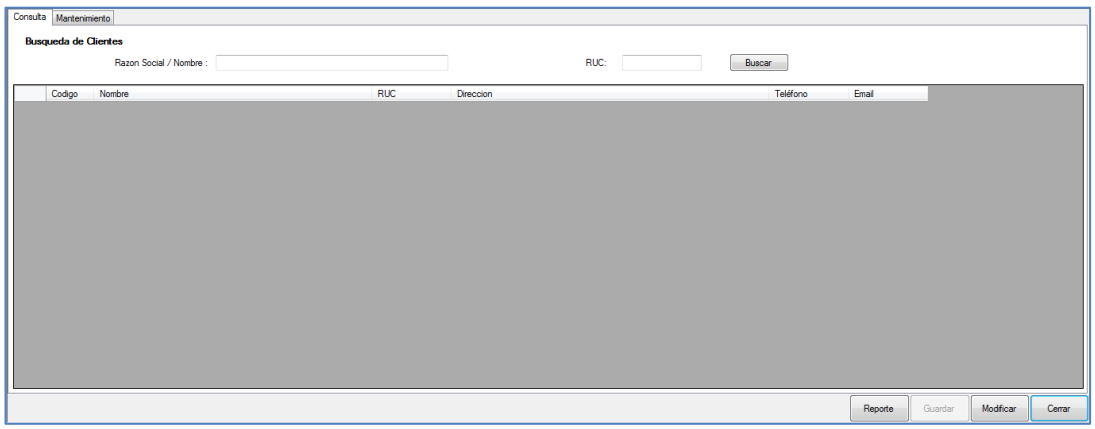


Gráfico 3.6 Formulario búsqueda de Clientes

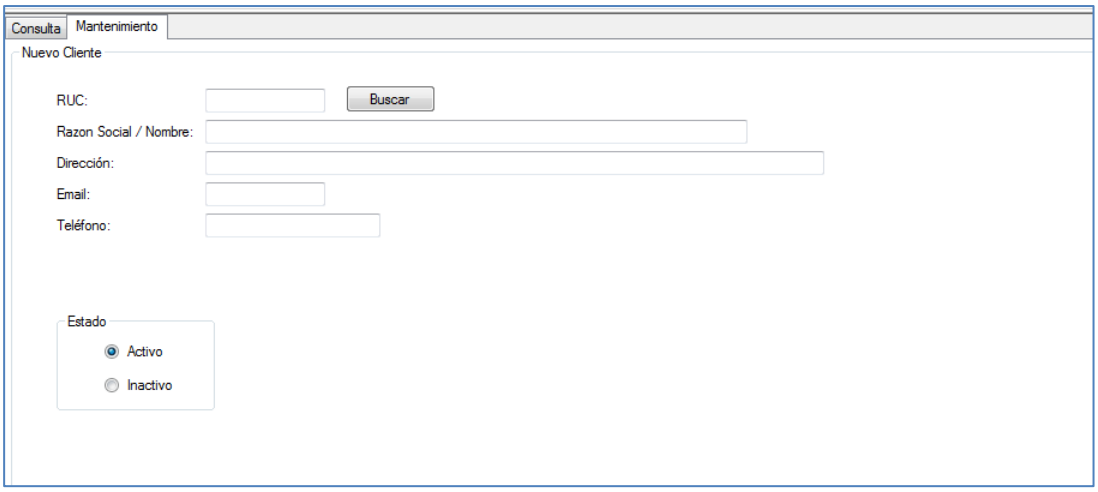


Gráfico 3.7 Formulario Registro/Modificación de Clientes

- Personal:

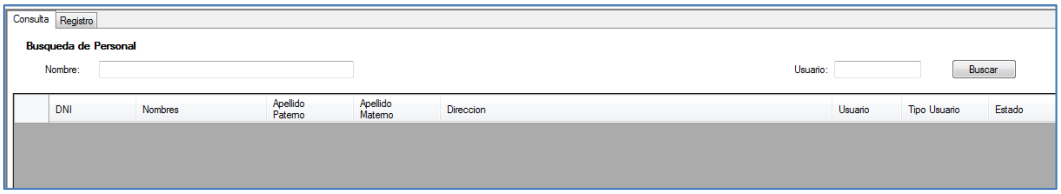


Gráfico 3.8 Formulario Búsqueda de Personal

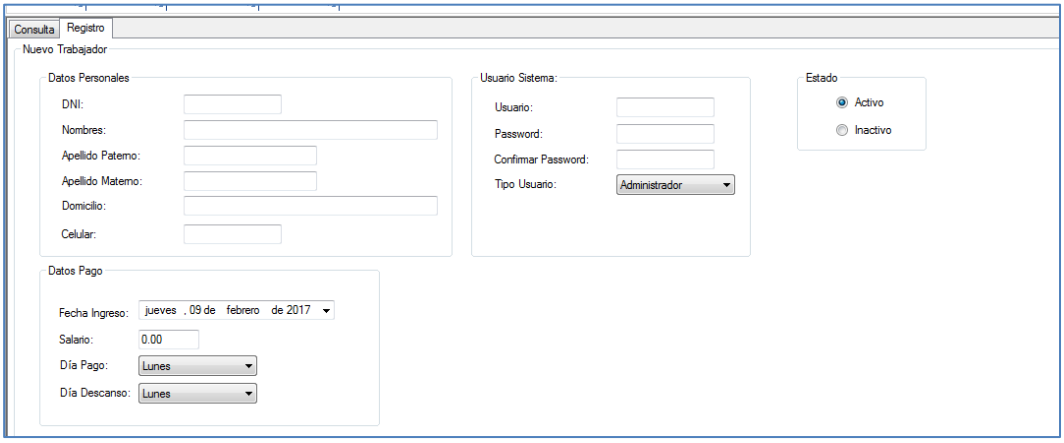


Gráfico 3.9 Formulario Registro de Personal

- Proveedores:

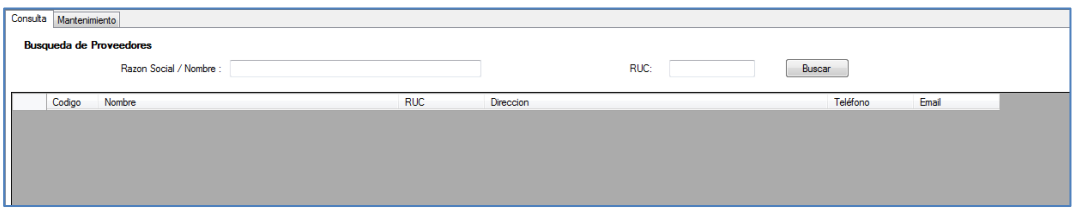


Gráfico 3.10 Formulario Búsqueda de Proveedores

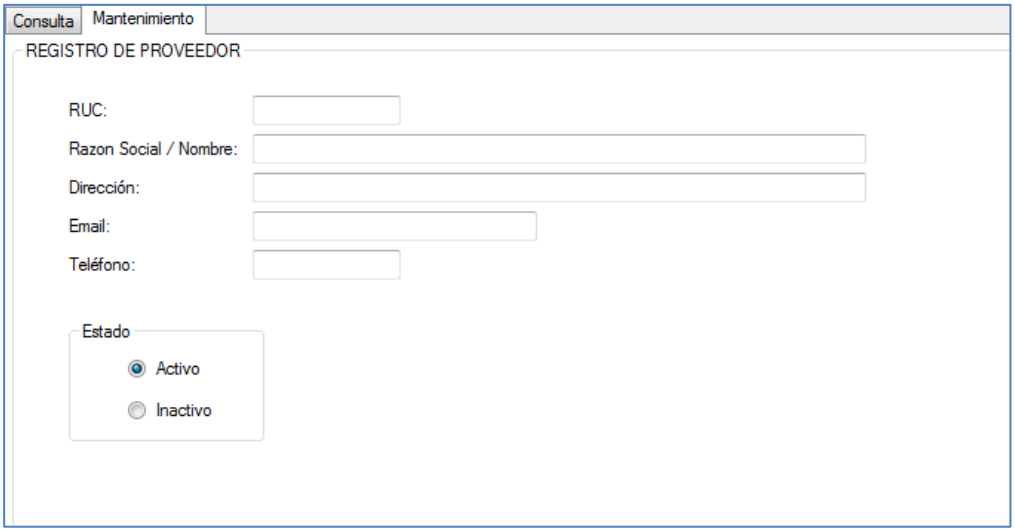


Gráfico 3.11 Formulario Registro de Proveedor

- Artículos:

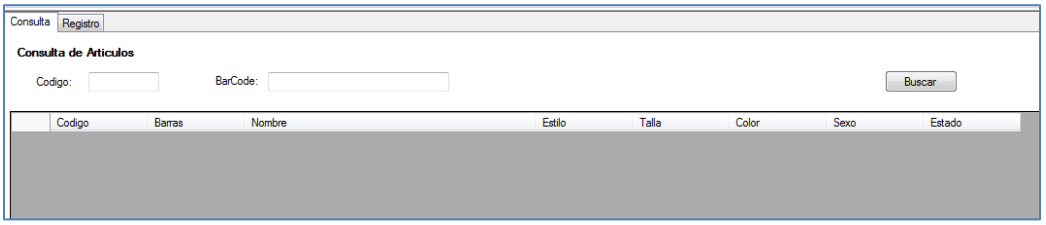


Gráfico 3.12 Formulario Búsqueda de Artículos

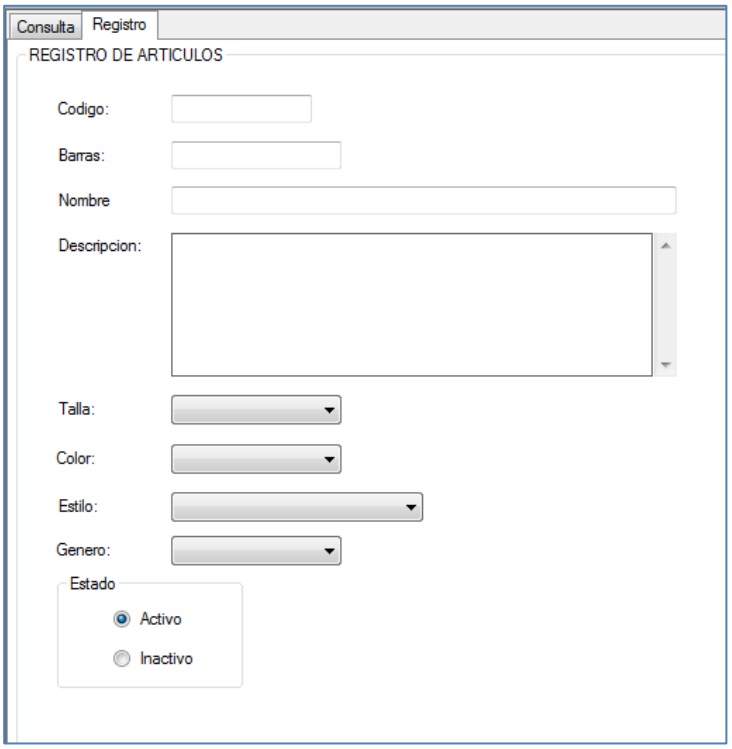


Gráfico 3.13 Formulario Registro de Artículos

Como característica principal podemos identificar que los formularios cuentan con 2 pestañas: Consulta y Mantenimiento, y en otros casos: Consulta y Registro. Se deben estandarizar los nombres, ya que dichos formularios tienen la misma función. (Registro y Mantenimiento)

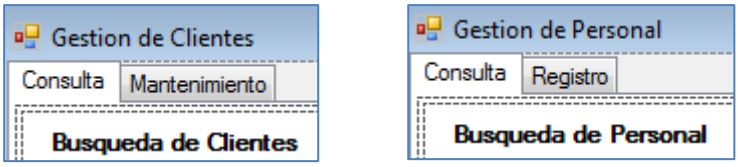


Gráfico 3.14 Diferencias en los nombres de los formularios

A continuación se presenta una tabla en la cual se identifican las entradas y salidas de cada proceso del sistema de Ventas y Facturación.

PROCESO	INPUTS	SALIDA
Búsqueda de Clientes	Razón Social / Nombre DNI	Listado de clientes
Registro de Clientes	RUC Razón Social / Nombre Dirección Email Teléfono Estado	Mensaje de confirmación de registro/modificación exitosa.
Búsqueda de Personal	Nombre Usuario	Listado de personal
Registro de Personal	DNI Nombres Apellido Paterno Apellido Materno Domicilio Celular Usuario Password Confirmar password Tipo Usuario Fecha Ingreso Salario Día Pago Día Descanso Estado.	Mensaje de confirmación de registro/modificación exitosa.

Búsqueda de Proveedores	Ruc Razón Social / Nombre	Listado de proveedores
Registro de Proveedores	Ruc Razón Social / Nombre Dirección Email Teléfono Estado	Mensaje de confirmación de registro/modificación exitosa.
Búsqueda de Artículos	Código BarCode	Listado de Artículos
Registro de Artículos	Código Barras Nombre Descripción Talla Color Estilo Genero Estado	Mensaje de confirmación de registro/modificación exitosa.

Tabla 5 - Entrada y Salida de cada Proceso del Sistema de Ventas

Según el análisis realizado identificamos y usando la información obtenida en la tabla, tenemos lo siguiente:

$$A = 42$$

$$B = 44$$

$$X = A / B$$

Donde, A = Número datos de entrada y salida identificados y B = total de datos de entrada y salida.

$$X = 42 / 44$$

$$X= 95\%$$

Los siguientes campos pueden traer problemas en la comprensión del usuario:

- Barras: se refiere el código de barras asignado al artículo, pero con el nombre asignado actualmente puede confundir al usuario.
- BarCode: nuevamente se refiere al código de barras, pero nuevamente el nombre no está claro.

3.1.2. Aprendizaje:

- ✓ **Efectividad del manual de usuario:**

En este caso el software evaluado no cuenta con un manual de usuario, por lo tanto, nuestra siguiente métrica no puede ser aplicada:

Efectividad del manual de usuario:

$$X = A / B$$

Donde, A = Tareas completadas con éxito después de usar el manual de ayuda y B = Total de funciones probadas.

- ✓ **Tiempo de aprendizaje de una función:**

T = Tiempo de aprendizaje.

Tomando como referencia el primer uso del software para realizar este análisis, obtenemos lo siguiente:

PROCESO	TIEMPO PROMEDIO	DIFICULTADES
Búsqueda de Clientes	20 seg	Ninguna
Registro de Clientes	2 min	Ninguna
Búsqueda de Personal	20 seg	Ninguna
Registro de Personal	5 min	En este formulario observamos que se tienen campos día de pago y día de descanso, los cuales tuvieron que consultarse debido a que no se entendía la definición.
Búsqueda de Proveedores	20 seg	Ninguna

Registro de Proveedores	2 min	Ninguna
Búsqueda de Artículos	1 min	El campo BarCode no está claro, posteriormente se explicó que refiere al código de barras de los artículos.
Registro de Artículos	5 min	Ninguna

Tabla 6 - Tiempos de Aprendizaje

3.1.3. Operatividad:

- ✓ **Número de funciones en el sistema que permiten ser canceladas:**

Anteriormente identificamos el total de funciones disponibles en el sistema, por lo tanto ahora nos toca verificar la capacidad de estas funciones para ser canceladas por el usuario.

Logramos identificar que todos los formularios cuentan con un botón cerrar, esto se debe a que el sistema mantiene un conjunto de botones como parte de la estructura para todos los formularios presentes.

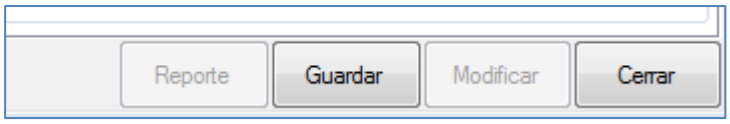


Ilustración 1

El botón cerrar en este caso cumple doble función:

1. Cerrar el formulario.
2. Cancelar un proceso, esto incluye un registro no culminado, una búsqueda o un reporte.

Por lo tanto:

$$X = A / B$$

$$X = 15/ 15$$

Donde, A = Funciones que pueden ser canceladas por el usuario, B = Total de funciones disponibles en el sistema. El 100% de las funciones disponibles pueden ser canceladas por el usuario.

3.2. Casos de Prueba – Capa de Presentación

Dado que en el sistema a rediseñar la capa de presentación no posee documentación, se procederá a realizar los Casos de Prueba de las principales funciones encontradas en el sistema.

3.2.1. Componente de Interfaz de Usuario:

El Software evaluado está usando la IGU Ribbon, compuesta de una banda (cintas) en la parte superior de una ventana donde se exponen todas las funciones que puede realizar un programa en un solo lugar.

Estas cintas están destinadas a mejorar la usabilidad, por la consolidación de las funciones del programa y los comandos en un lugar fácilmente reconocible, no es necesario mirar a través de múltiples niveles jerárquicos de los menús, las barras de herramientas o paneles de tareas antes de encontrar el comando.

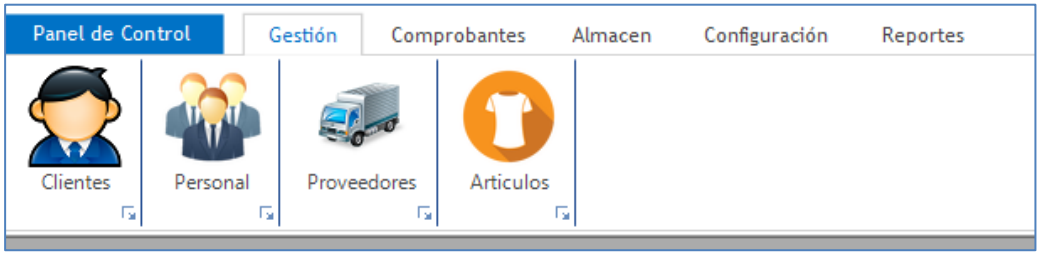


Ilustración 2

Básicamente la aplicación muestra iconos grandes con sus respectivos títulos, cada título identifica una funcionalidad disponible en el sistema.

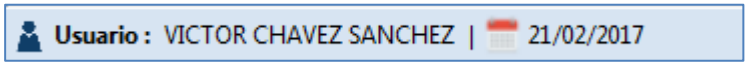


Ilustración 3

En la parte inferior únicamente tenemos 2 datos, el nombre del usuario "logeado" en el sistema y la fecha.

También notamos que el botón de Panel de Control no se está usando, sería mejor retirarlo y así ganar más espacio, esto también ayudaría a evitar confusiones en el usuario al tratar de acceder en dicha opción:

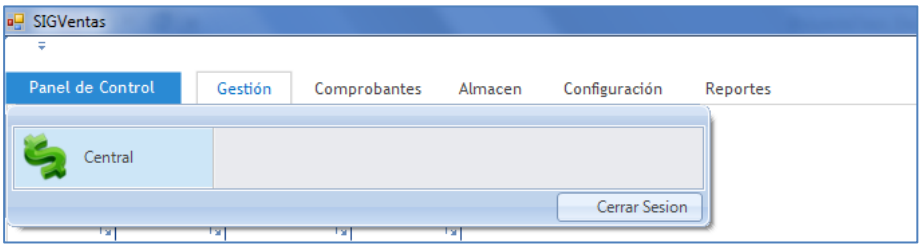


Ilustración 4

A continuación comenzaremos a aplicar nuestros indicadores definidos en este proyecto:

- ✓ **Incompatibilidad en la resolución de pantallas:**

La manera más práctica y sencilla de comprobar la compatibilidad de las pantallas en las distintas resoluciones del sistema es cambiar la resolución de manera manual de nuestras laptops o computadoras de escritorio.

Los resultados de las pruebas son las siguientes:

- ➢ **800 x 600:** Hay que tener en cuenta que esta resolución ya casi no es usada, salvo equipos muy antiguos, por lo tanto se considera que los resultados en esta resolución no podrían considerarse como críticas.

  En el Menú principal observamos que no hay problemas en la resolución, todas las opciones se visualizan perfectamente.

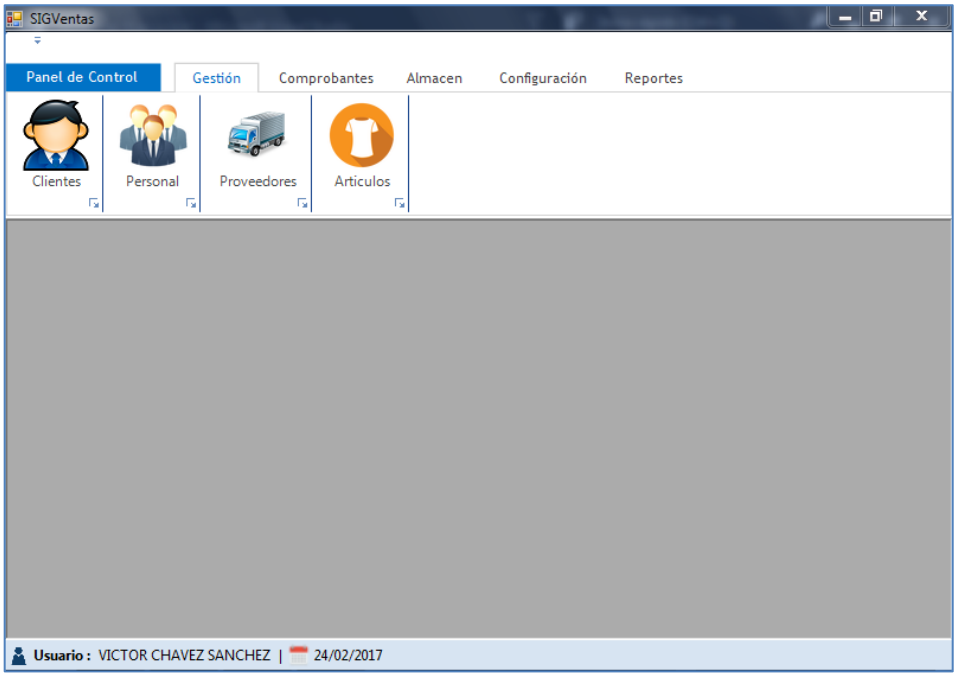


Ilustración 5 – Imagen 1 – Menu principal resolución 800x600

Entrando más en las funciones del sistema (Gestión de Clientes) nos encontramos con el primer inconveniente (ver Imagen 2):

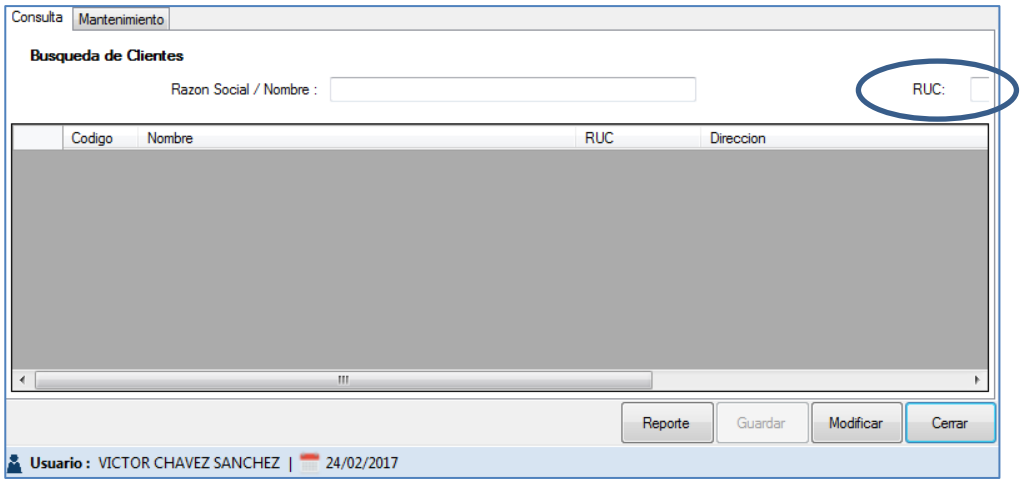


Ilustración 6 - Imagen 2 – Pantalla "Consulta", campo RUC no se visualiza completamente

El campo "RUC" en la búsqueda de clientes no se logra apreciar en su totalidad, lo que no limita a solo usar el nombre para la búsqueda, lo cual no es correcto.

Así mismo en la parte de registro de Cliente (Mantenimiento) podemos notar que el campo estado a las justas logra visualizarse en su totalidad:

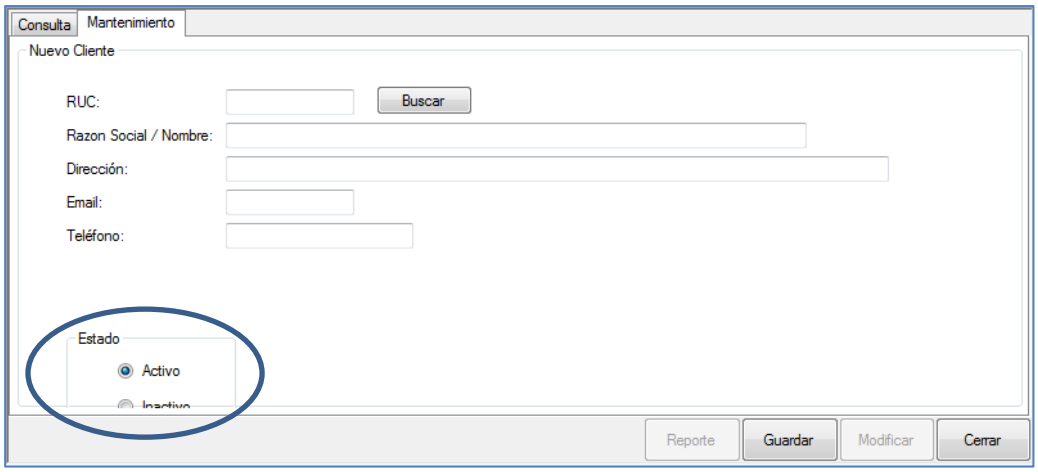


Ilustración 7 – Radio Buttom fuera del rango

Para el segundo análisis continuamos con el formulario Gestión de Personal, al parecer no hay problemas en la visualización.

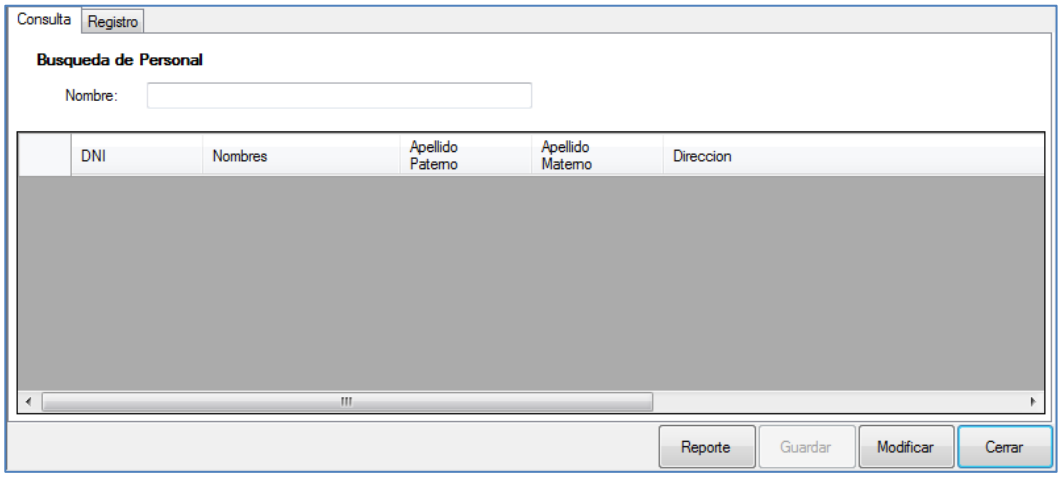


Ilustración 8 – Pantalla formulario Gestión de Personal

Al ubicarnos en la pestaña “Registro” nos encontramos con los campos totalmente ajustados y sin poder visualizarlos adecuadamente.

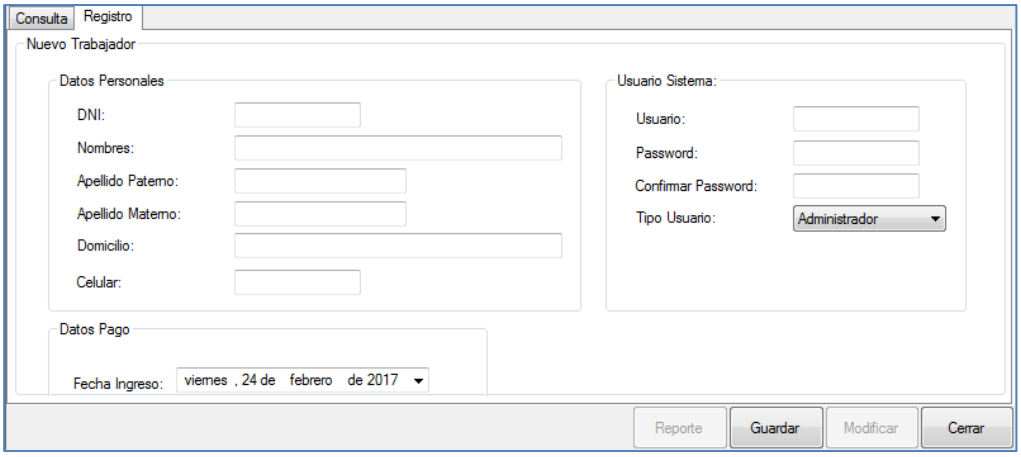


Ilustración 9 – Pantalla de Registros de Personal con problemas de visualización.

Definitivamente el software no está apto para ser usado en la resolución 800 x 600, considero que no es necesario realizar más pruebas.

- **1366 x 768:** Esta resolución es la más común en la mayoría de computadoras de escritorio y laptops, por lo que los resultados en estas pruebas si pueden considerase muy importantes.

  En el menú principal, al igual que en la anterior prueba, no encontramos ningún problema en la visualización.

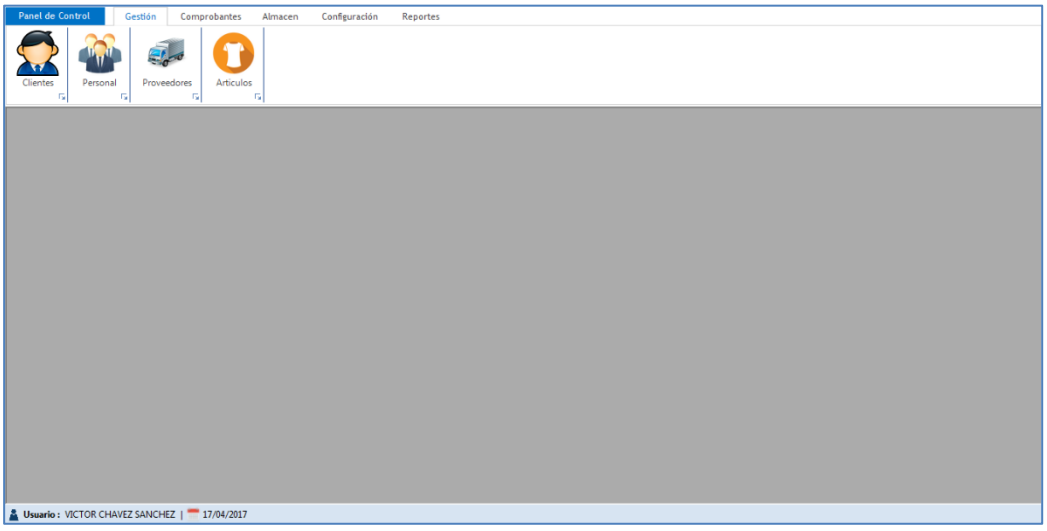


Ilustración 10 – Pantalla Principal del software, se visualiza sin problemas.

Navegando en la pestaña Consulta del formulario Gestión de Clientes nos encontramos que no hay problemas en la visualización, el campo RUC se visualiza en su totalidad.

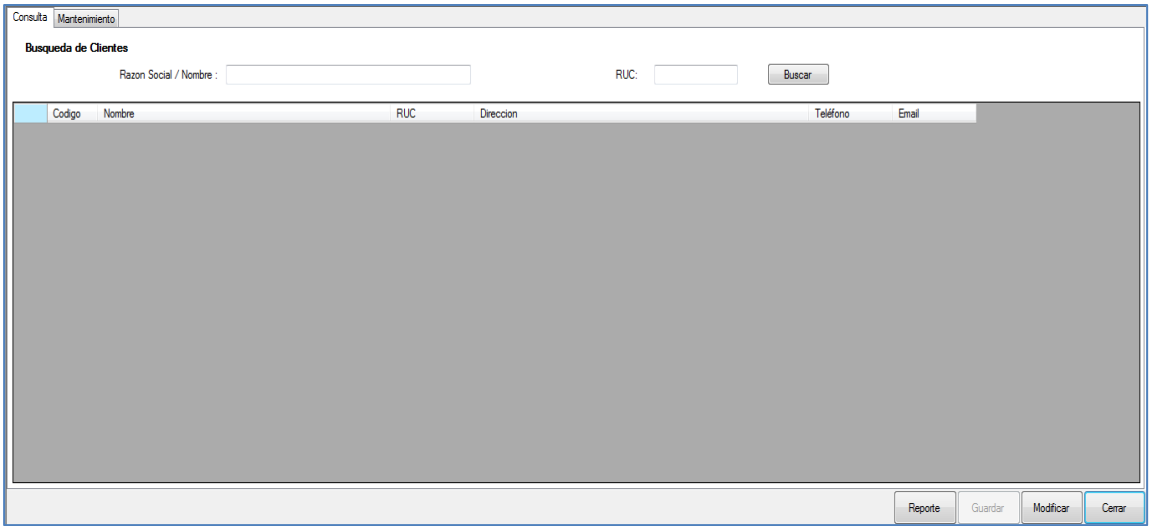


Ilustración 11 – Formulario de Consultas en Gestión de Clientes, se visualiza sin problemas.

Navegando en la siguiente pestaña: "Mantenimiento", también se visualizan todos los campos de manera adecuada, incluso podemos indicar que sobra mucho espacio en el formulario.

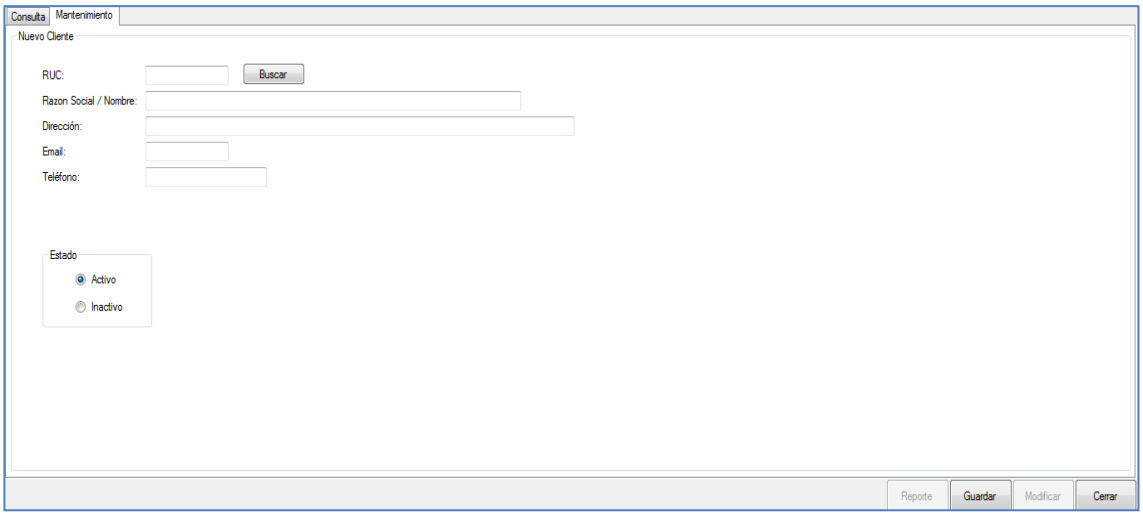


Ilustración 12 – Pestaña "Mantenimiento" en formulario Gestión de Clientes.

A continuación se procede con el formulario Gestión de Personal, de igual manera la pestaña “Consulta” se aprecia de manera adecuada.

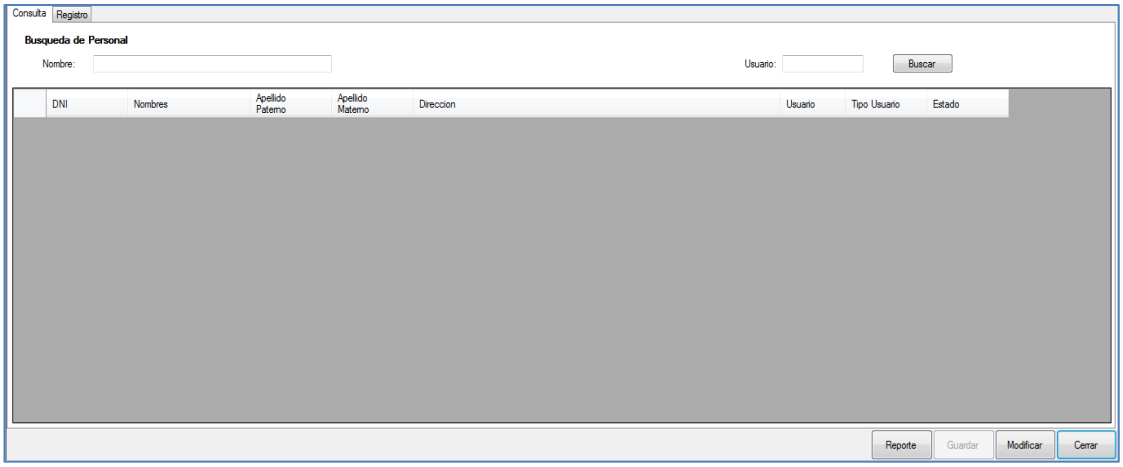


Ilustración 13 – Pantalla del Formulario Gestión de Personal, pestaña “Consulta”.

De igual forma, en la pestaña “Registro” todos los campos se visualizan correctamente en la pantalla.

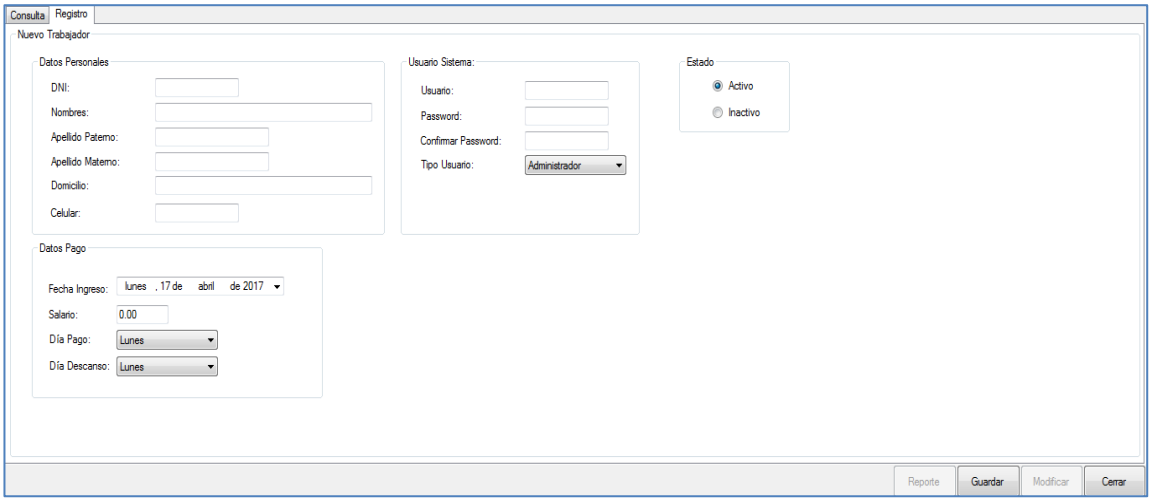


Ilustración 14 – Pantalla del Formulario Gestión de Personal, pestaña “Registro”

De manera general no existen problemas en los formularios al usar la resolución 1366 x 768, considero que no es necesario revisar el resto de formularios después de los resultados obtenidos.

- **1024 x 1080:** Esta resolución al ser mayor que la anterior no debería causar problemas en la visualización de los formularios, por lo tanto no se evaluará.

- ✓ **Reutilización de controles en la interfaz:**

En los formularios se han reutilizado la botonera inferior, dichos botones aplican para cualquier tipo de formulario.

En este caso la botonera repite código por cada formulario, pero no en cada formulario, es decir, se está reutilizando esa misma botonera para realizar las siguientes funciones:

-Cerrar

-Registro

-Modificación

-Exportación

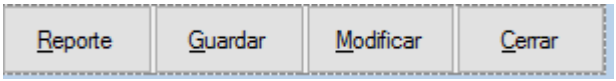


Ilustración 15 - Botonera estándar parte del diseño en el software.

Como comente anteriormente, este diseño es estándar en todo el Software por lo que no precisa de cambio alguno.

- ✓ **Organización de los elementos del formulario:**

En esta parte del informe se verificara que los elementos estén organizados de manera adecuada, para lo cual todos los formularios deben seguir un patrón definido.

- **Login:** Los campos están alineados de manera correcta, así mismo la posición de los botones es adecuada.

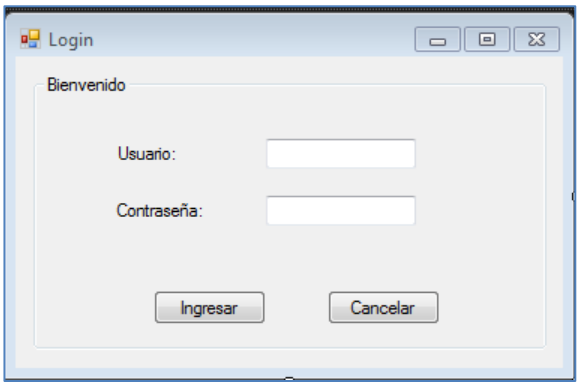


Ilustración 16 – Formulario Inicio de Sesión

- **Menú Principal:** En este caso se usa Ribbon, los botones grandes están alineados de manera correcta, y divididos por cada pestaña, al ser pocos los botones el tamaño de los botones es adecuado.

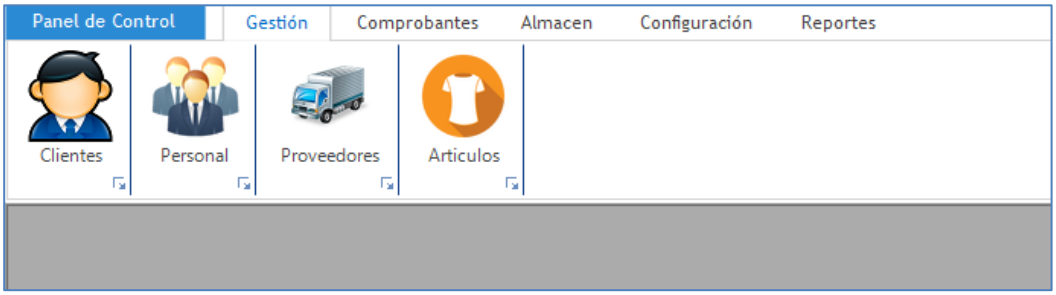


Ilustración 17 – Formulario Menú Principal

- **Gestión Clientes:** El formulario cuenta 2 pestañas: Consulta y Mantenimiento, para el caso de la pestaña “Consulta” encontramos 2 campos para la búsqueda de clientes: Nombre/Razón Social y Ruc.

Los botones en la parte inferior están alineados y con el tamaño adecuado para ser visibles al usuario

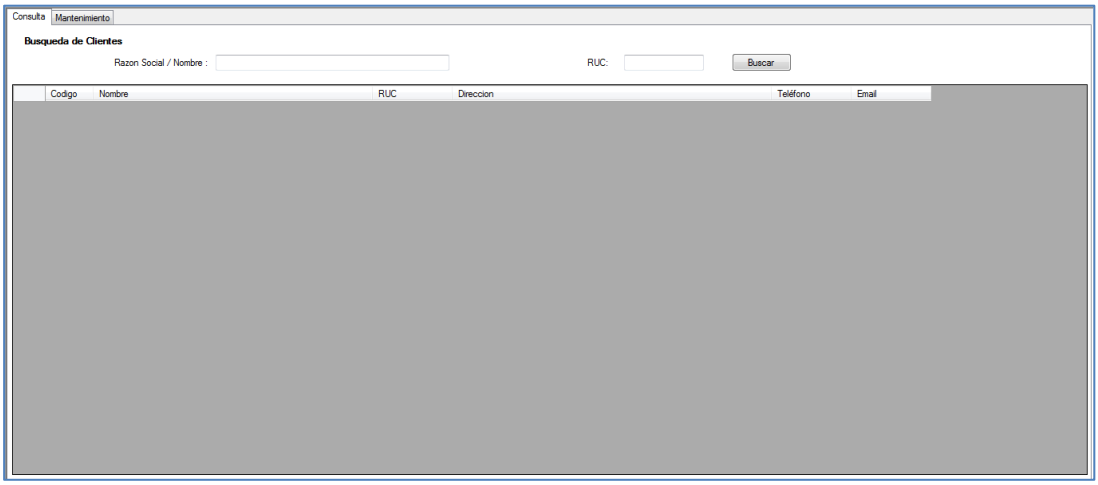


Ilustración 18 – Pantalla de "Consulta" en Gestión de Clientes.

Ubicándonos en la pestaña mantenimiento encontramos que los campos están alineados, los "textbox" están de diferentes tamaños, lo adecuado es que, según las definiciones, se mantenga el mismo largo. También se observa mucho espacio libre, la tecla "TAB" no nos lleva entre los campos en el orden adecuado. (Desde el campo RUC nos lleva a Email).

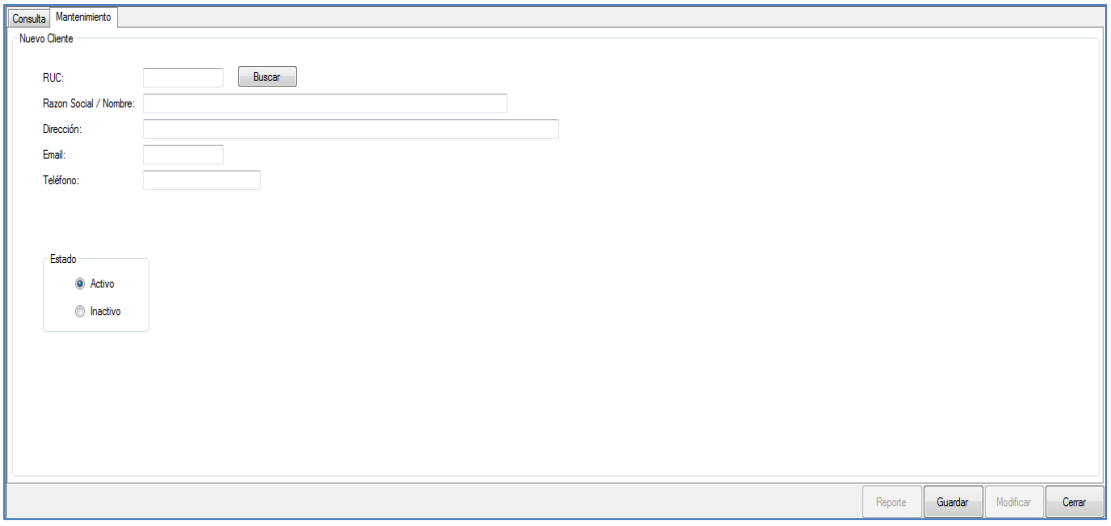


Ilustración 19 – Pantalla "Mantenimiento" en Gestión de Clientes

Como primer ordenaremos de manera correcta la función de la tecla TAB, entraremos a la opción a través del menú

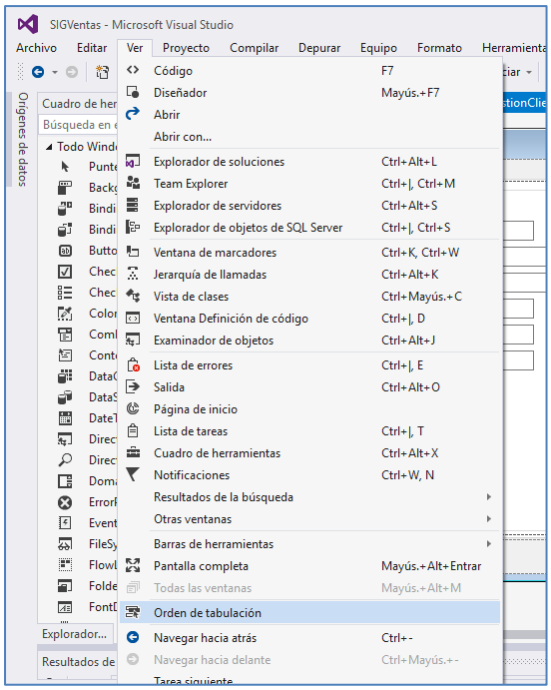


Ilustración 20 - Orden de Tabulación, Visual Studio

A continuación nos mostrara el orden actual del cursor al usar la tecla “Tab”. Claramente podemos observar que no están en orden alguno.

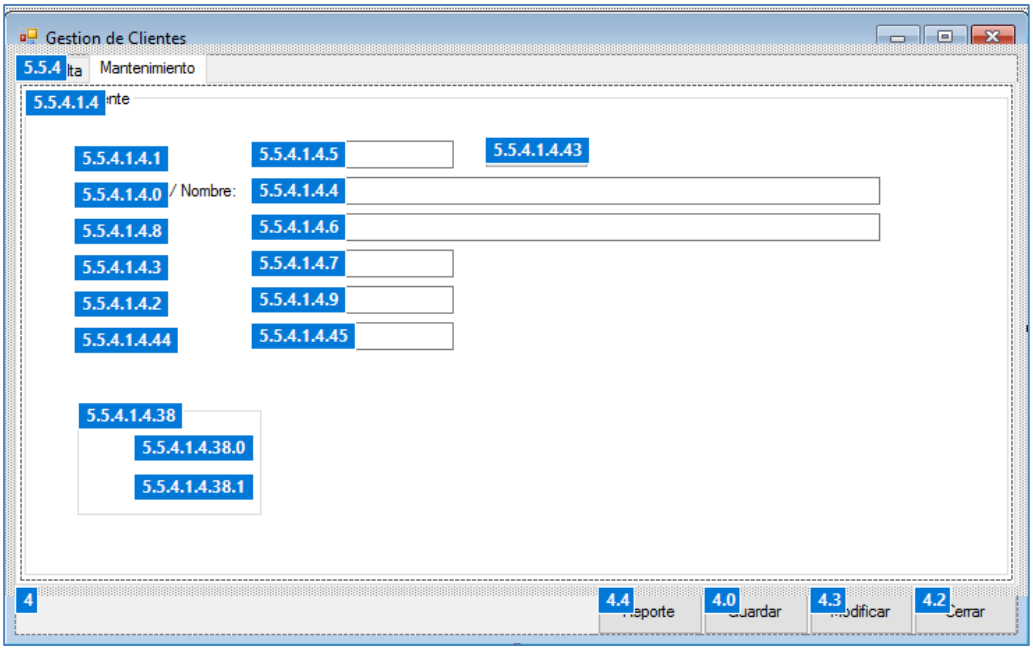


Ilustración 21 - Visualización del orden actual del puntero de selección.

Visual Studio nos facilita toda esta acción a solo dar clic cada componente según el orden en el que se moverá el puntero de "Selección de texto".

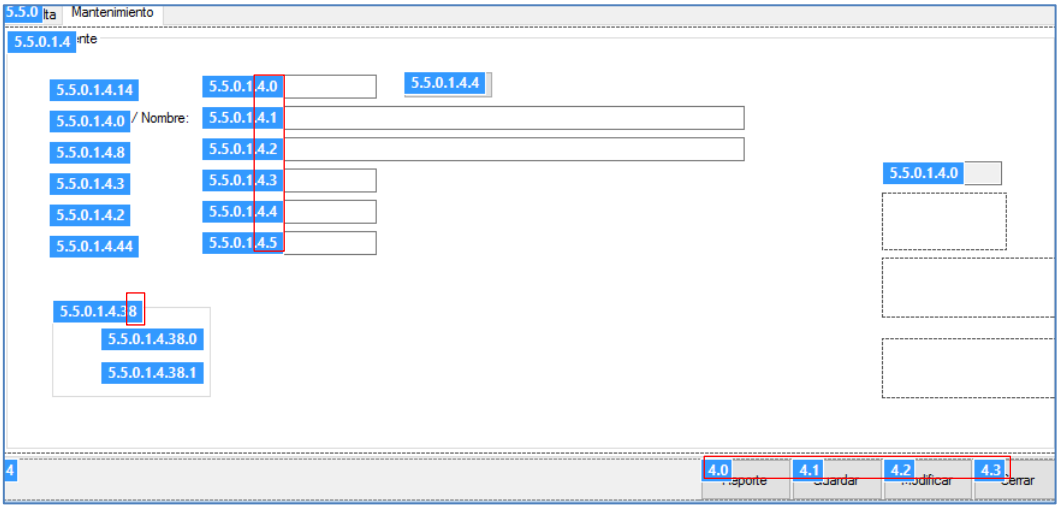


Ilustración 22 - Nuevo orden de secuencia de la Tecla "Tab".

Concluida la configuración del orden correcto del tabulador, continuamos con la configuración de teclas abreviadas, las teclas abreviadas son de suma importancia ya que ayuda a el usuario de no depende del "Mouse", así mismo no todas las estaciones de trabajo tendrán disponible dicho dispositivo.

Básicamente las abreviaturas se aplican en los botones y en las pestañas, que son las opciones que más requieren del uso del "Mouse".

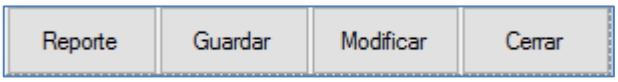


Ilustración 23 - Botones principales del formulario.

Esta configuración es rápida y sencilla, únicamente tenemos que agregar el símbolo "&" delante del texto asignado a cada control.

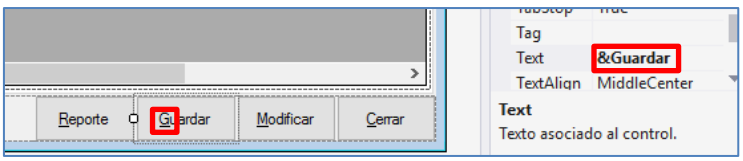


Ilustración 24 - Asignación de teclas rápidas a Botonera

Con estos cambios nuestro formulario quedaría mucho mejor organizado y agradable a la vista.

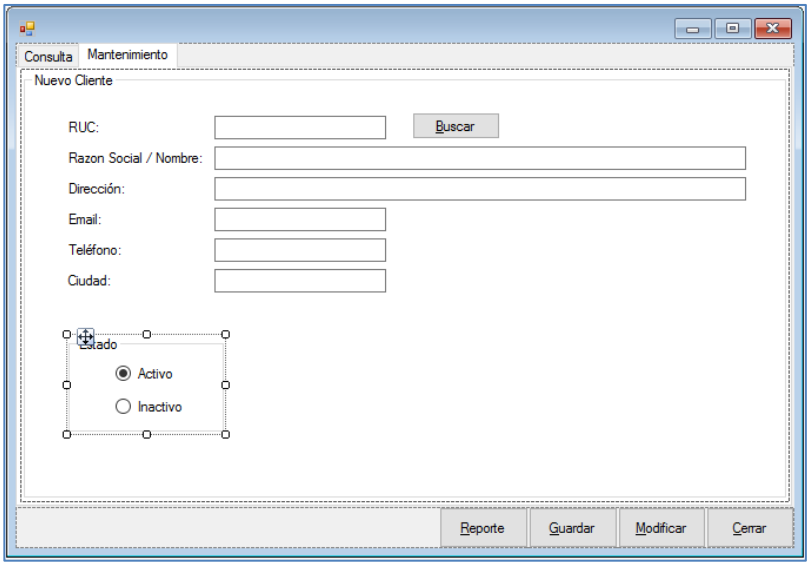


Ilustración 25 - Formulario Gestión de Clientes propuesto.

- **Gestión de Personal:** El formulario cuenta con 2 pestañas: Consulta y Registro, ubicándonos en la pestaña “Consulta” encontramos 2 campos para la búsqueda de clientes: Nombre y Usuario.

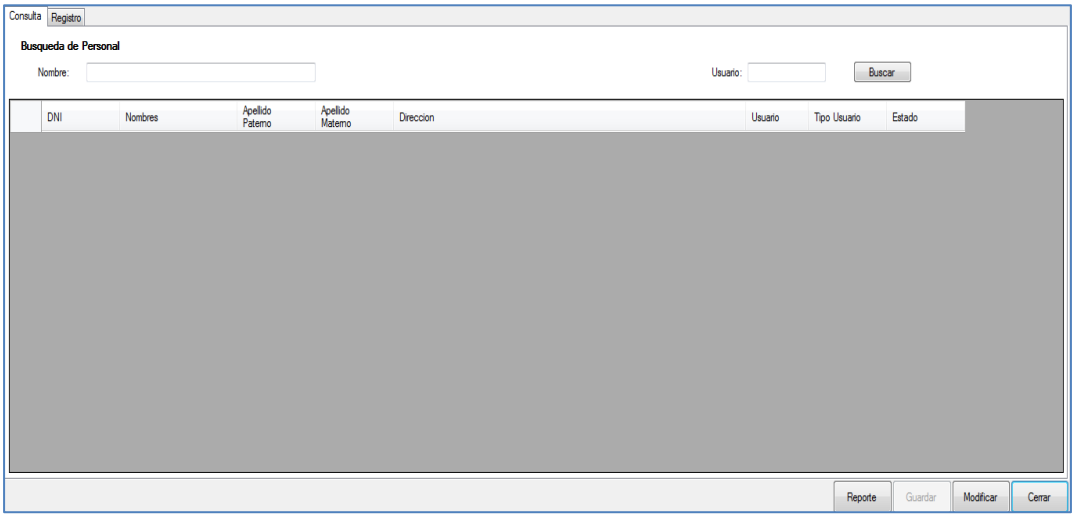


Ilustración 26 – Pantalla “Consulta” en Gestión de Personal

Los botones en la parte inferior están alineados y con el tamaño adecuado para ser visibles al usuario, sería apropiado agregar un campo de búsqueda a través del número de DNI. (Ver ilustración 26)

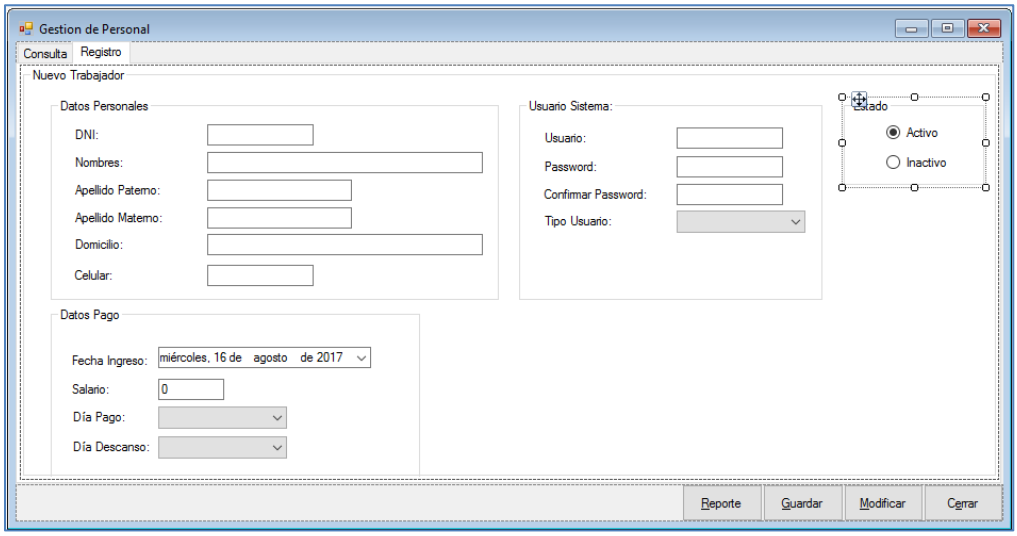


Ilustración 27 – Pantalla “Registro” en Gestión de Personal

En la parte de registro del mismo formulario nuevamente encontramos que el largo de los campos en el formulario es disparejo. La tecla “TAB” no nos lleva a los campos en el orden adecuado. (Ver ilustración 27).

En la pestaña Consulta se agregaron las teclas rápidas en los botones así mismo la búsqueda ya no se realizará por Usuario, lo ideal es que se realice por DNI, que es el identificador universal de las personas. Aplicaremos todas las acciones aplicadas en el formulario “Gestión Clientes”, recordemos que todo debe estar estandarizado para todas las pantallas. (Ver ilustración 28)

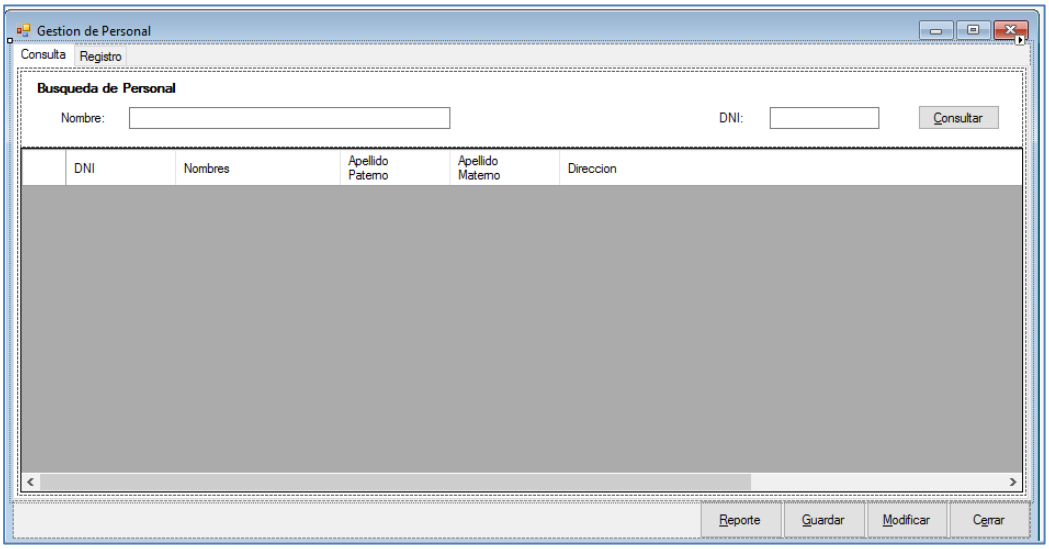


Ilustración 28 - Botones con tecla rápida, búsqueda por DNI.

En la pestaña Registro organizaremos los controles, iniciando con mantener el largo de los “TextBox”, luego reorganizando la función del Tabulador.

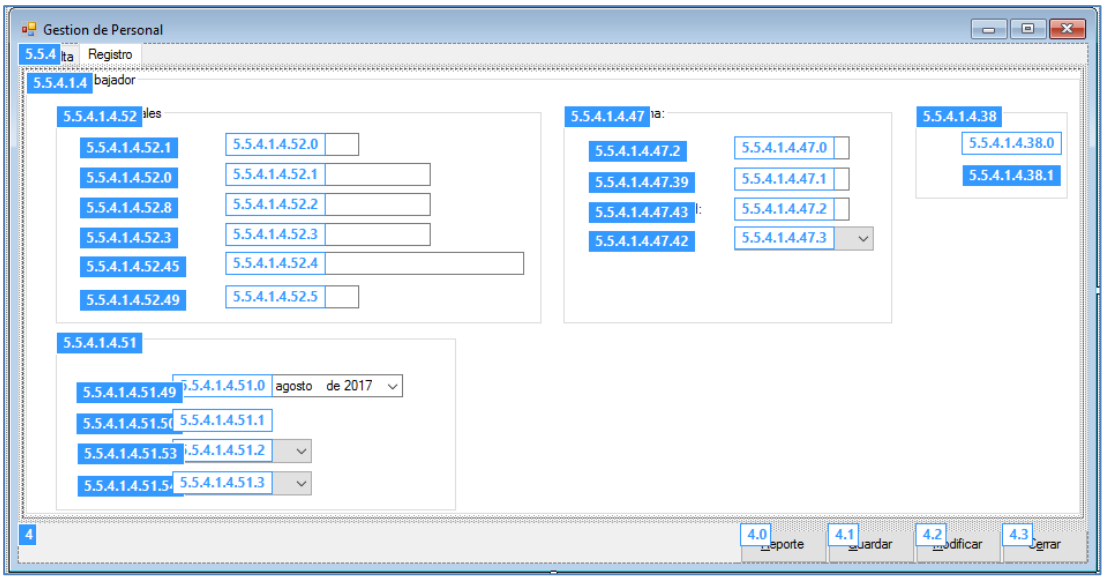


Ilustración 29 - Reorganización del tabulador.

Realizados los cambios, el formulario final propuesto sería el siguiente:

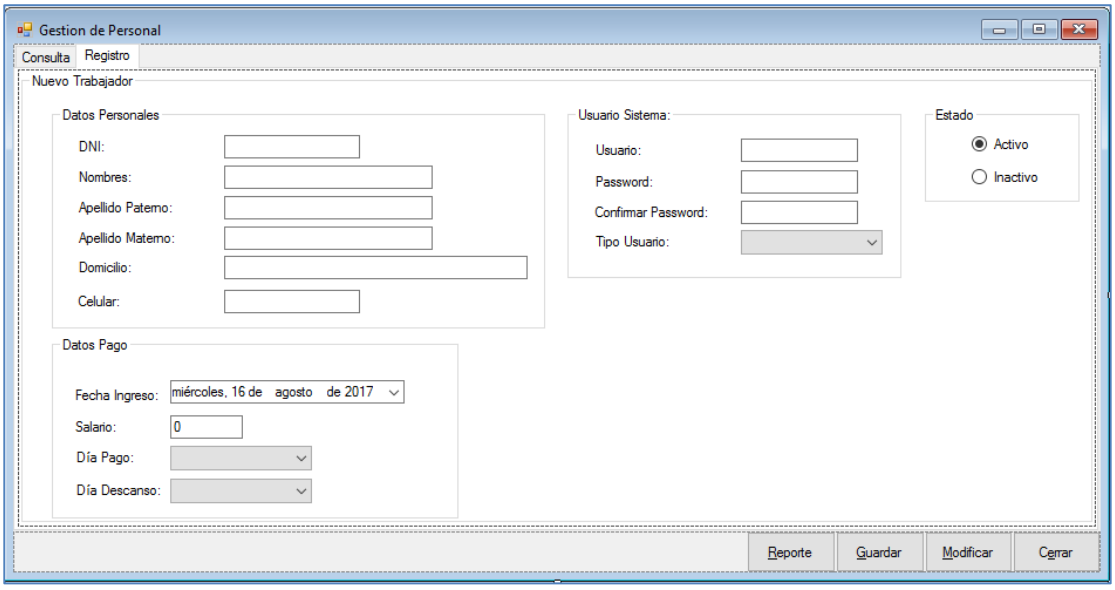


Ilustración 30 - Formulario "Gestión de Personal" propuesto.

- **Gestión de Ventas:** El formulario cuenta con 2 pestañas: Consulta y Ventas, ubicándonos en la pestaña “Consulta” encontramos 2 campos para la búsqueda de clientes: Correlativo y Nro. De Documento.

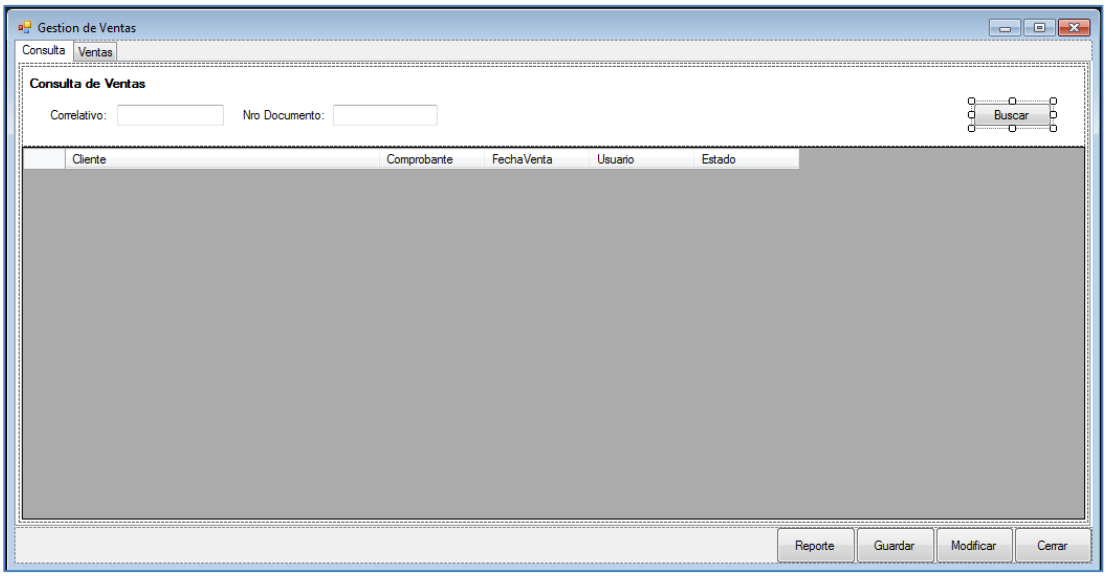


Ilustración 31 - Pantalla “Consulta” en Gestión de Ventas

El campo “correlativo” no es adecuado, la búsqueda siempre se realiza por número de documento, falta agregarse el filtro tipo documento de venta, no existen teclas abreviadas. (Ver ilustración 31)

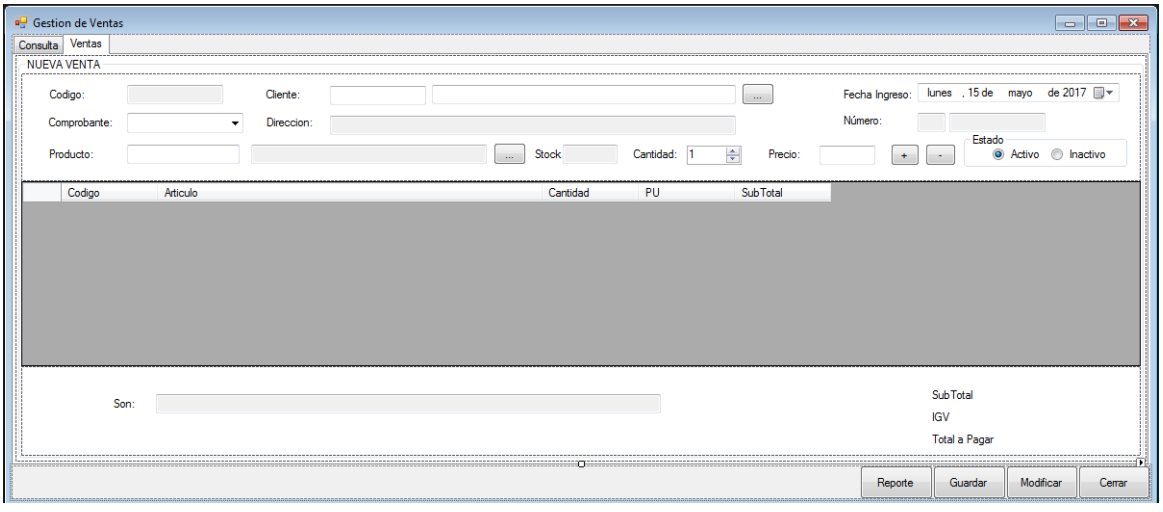


Ilustración 32 - Pestaña "Venta" en Gestión de Ventas

En la pestaña “Ventas” no existen teclas abreviadas, así mismo la tecla “TAB” no nos está moviendo entra los campos en el orden establecido. Tenemos que resaltar los campos más importantes.

El botón Buscar se mantiene en su posición a pesar de que la pantalla sea más grande o más pequeña, esto se soluciona con la propiedad “Anchor” lo que permitirá que el control se “enganche” a un lado del formulario moviéndose en conjunto con él sin importar la resolución de pantalla.

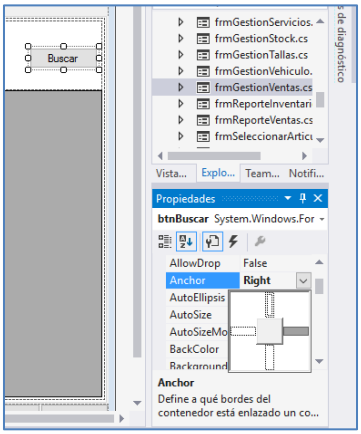


Ilustración 33 - Propiedad "Anchor" aplicada a botón "Buscar"

Por último organizamos los campos para búsqueda “Nro Documento” y se agregó la búsqueda por Tipo de Documento. A su vez se agregaron las teclas rápidas en esta primera pestaña.

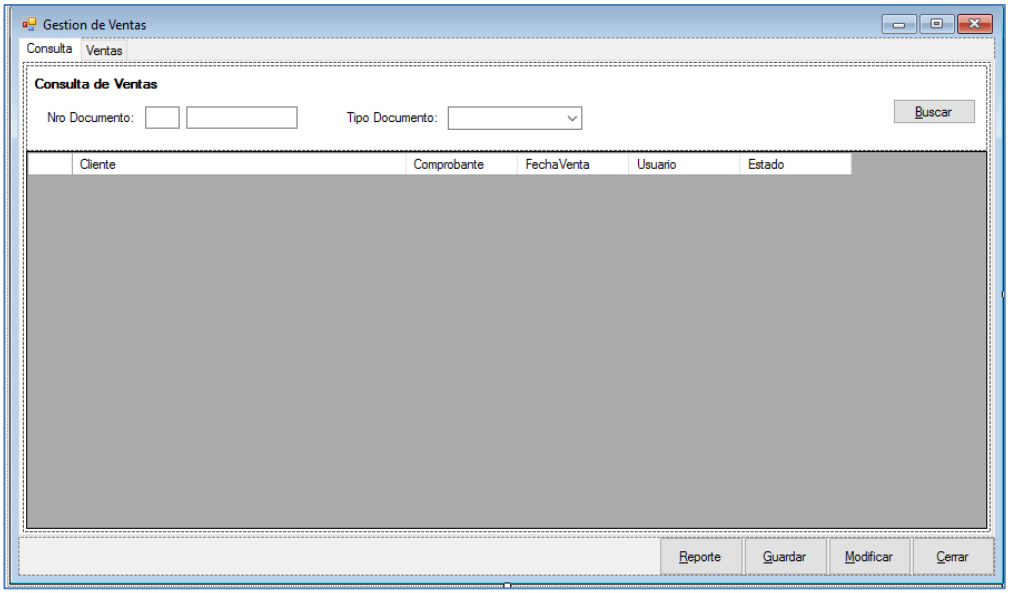


Ilustración 34 - Pestaña Consulta del formulario "Gestión de Ventas"

Siguiendo con la pestaña “Ventas” del mismo formulario, el campo “Código” no es un campo que sea de interés para los usuarios finales, entendemos que este código refiere a la codificación interna de las ventas, por lo tanto será retirado del formulario.

Se aumentó el tamaño de texto de los campos: número de documento, fecha de ingreso y los montos de las ventas, campos de suma importancia en una venta. Por último la organización del tabulador.

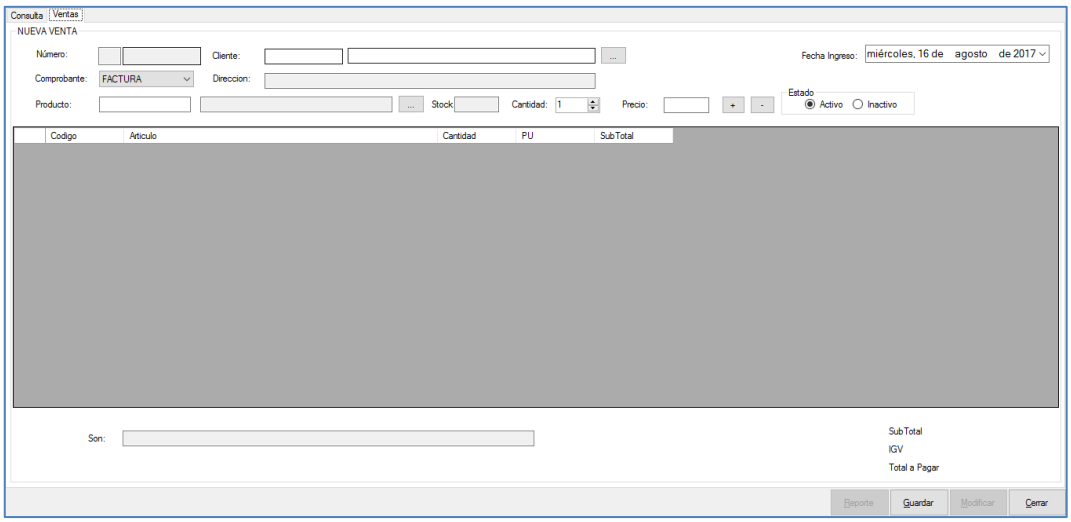


Ilustración 35 - Propuesta de la pestaña "Ventas" del formulario Gestión de Ventas

- **Gestión de Ingresos:** El formulario cuenta con 2 pestañas: Consulta e Ingresos Almacén, ubicándonos en la pestaña “Consulta” encontramos 1 campo para la búsqueda por proveedores.

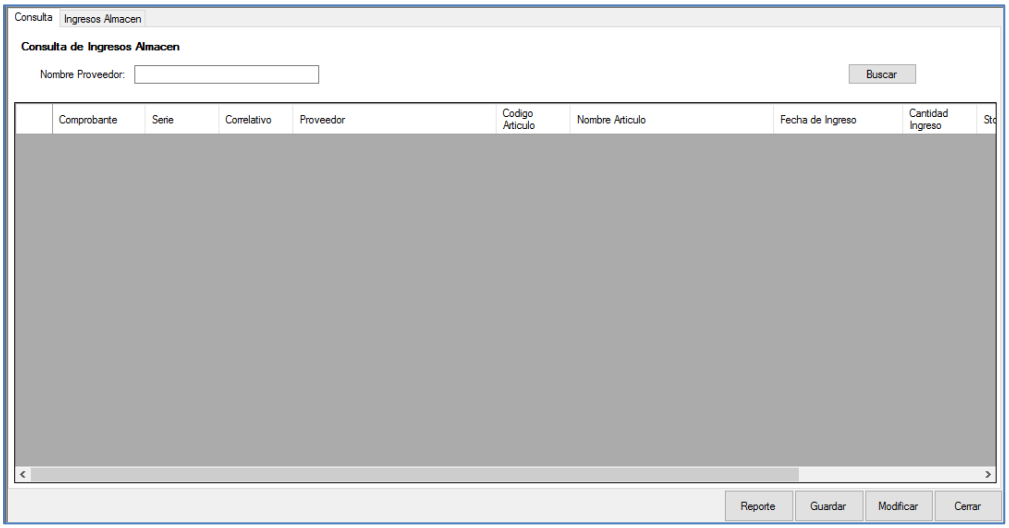


Ilustración 36 - Pantalla "Consulta" del formulario Ingresos Almacén

Solo existe la búsqueda por Proveedor, falta agregarse el filtro por fecha y tipo de comprobante, los botones no tienen tecla abreviada. (Ver ilustración 36)

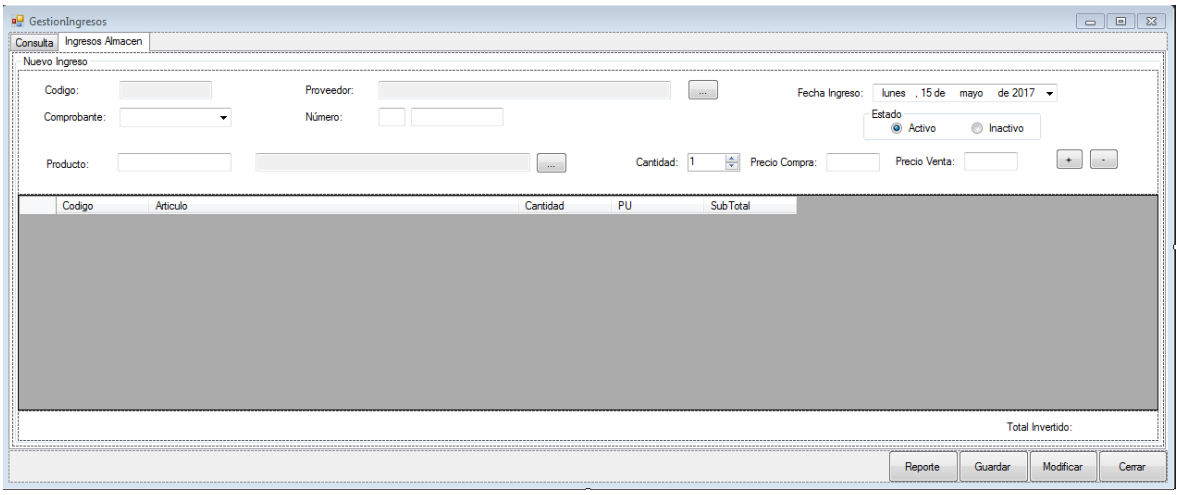


Ilustración 37 - Pantalla "Ingresos" de Formulario Gestión de Ingresos

En la pestaña “Ingresos Almacén” los controles no están organizados de la manera correcta, así mismo el espacio no está bien aprovechado. Del mismo modo están ausentes las teclas abreviadas y se muestran campos que no son de interés para el usuario final, en este caso se visualiza un campo “Código”. (Ver ilustración 37)

En el formulario encontramos 2 botones con descripción “…” si bien es cierto la mayoría de usuarios que ya han trabajo con un sistema informativo anteriormente ya conocen que refiere a la “búsqueda” de algún ítem, también hay usuarios nuevos que no saben a qué refiere; para estos casos no se modificar la descripción del botón pero si se agregarán “tooltips” para ayudar al usuario a entender la función específica de cada control; esto también aplicará para los botones “+” y “-“. (Ver ilustración 38)

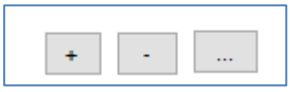


Ilustración 38 - Botones sin descripción especifica.

Nuestro fragmento de código para el diseño de “tooltips” sería el siguiente:

```
private void myToolTip()
{
        ToolTip myToolTip = new ToolTip();
        myToolTip.SetToolTip(btnBuscarProducto, "Buscar
        Producto");
        myToolTip.SetToolTip(btnBuscarProveedor, "Buscar
        Proveedor");
        myToolTip.SetToolTip(btnAgregar, "Agregar Ingreso");
        myToolTip.SetToolTip(btnQuitar, "Quiter Ingreso");
}
```

Con este fragmento de código configuramos algunos botones (también aplica para otros controles) para mostrar texto cuando el curso este encima de este.

Agregamos todas las características incluidas anteriormente en el resto de formularios y la propuesta sería la siguiente:

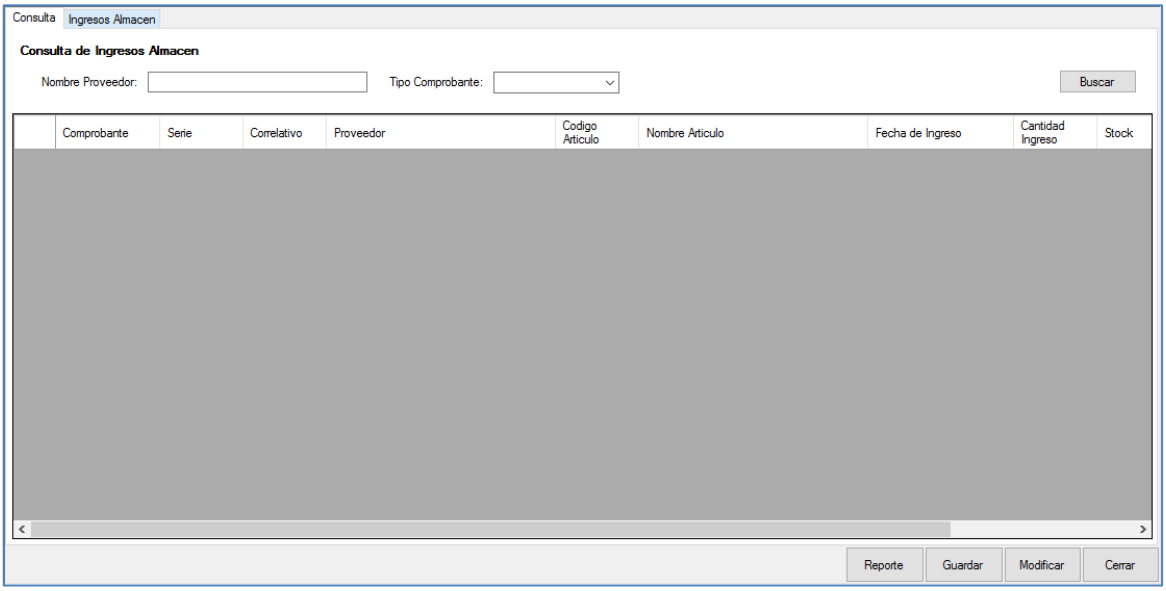


Ilustración 39 - Pestaña "Consulta" del formulario Gestión de Ingresos propuesta.

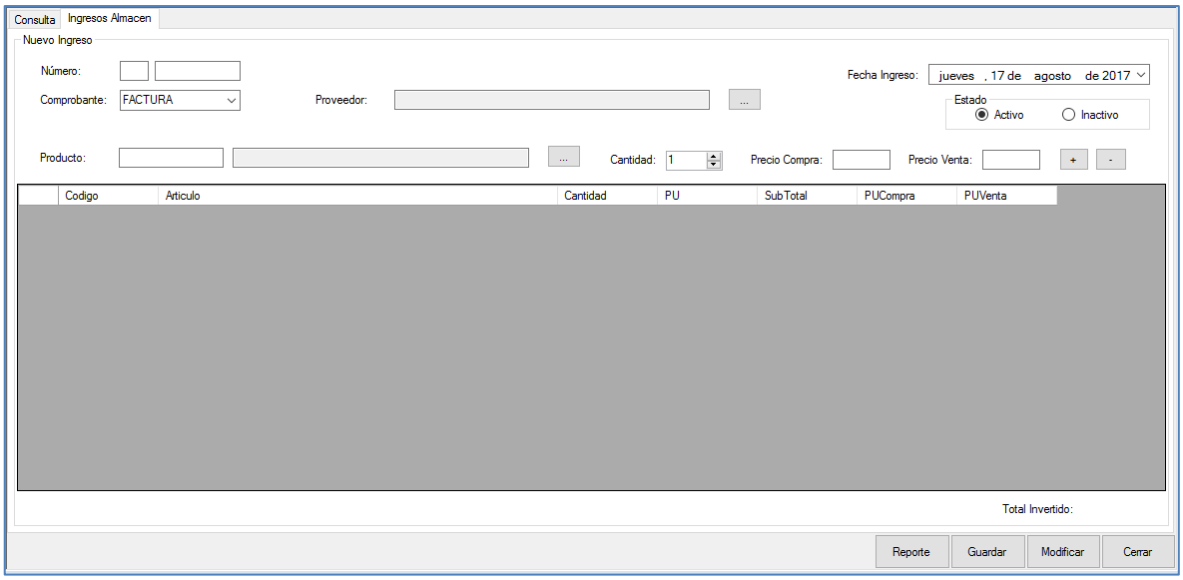


Ilustración 40 - Pestaña "Ingresos Almacen" en Gestión de Ingresos

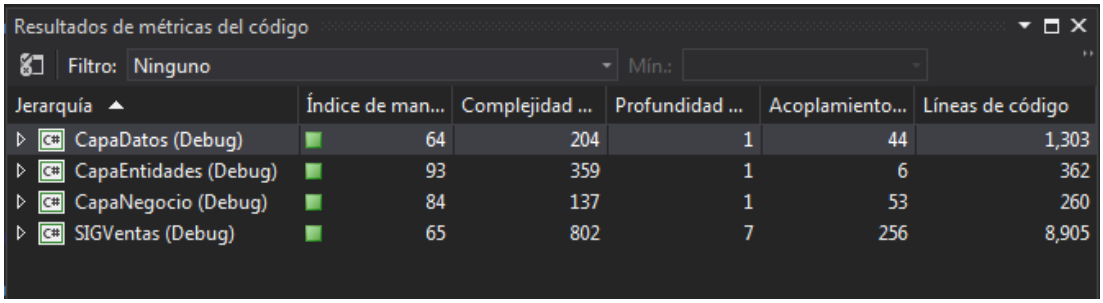
Debido a que el resto de formularios tienen en su mayoría las mismas características se obviaran en el presente informa.

3.2.2. Componente de Lógica de Presentación:

Visual Studio nos ofrece una herramienta interesante para los desarrolladores, hablamos de: “Calcular métricas de código para la solución”.

Las métricas de código son un conjunto de medidas de software que proporcionan a los programadores una mejor visión del código que están desarrollando. Al aprovechar las métricas de código, los programadores pueden entender qué tipos y métodos se deben rehacer o probar más a fondo. Los equipos de desarrollo pueden identificar los riesgos potenciales, entender el estado actual de un proyecto y seguir el progreso durante el desarrollo del software.

Para este análisis nos centraremos únicamente en la Capa de Presentación (SIGVentas), según lo arrojado por los resultados de las métricas obtenemos:



Jerarquía	Índice de man...	Complejidad ...	Profundidad ...	Acoplamiento...	Líneas de código
CapaDatos (Debug)	64	204	1	44	1,303
CapaEntidades (Debug)	93	359	1	6	362
CapaNegocio (Debug)	84	137	1	53	260
SIGVentas (Debug)	65	802	7	256	8,905

Ilustración 41 - Resultados de métricas del código del proyecto a rediseñar

Esta imagen inicial nos servirá para ver, en base a los cambios que realicemos, poder realizar un comparativo de las métricas obtenidas antes y después del rediseño de la Capa de Presentación.

Así mismo para complementar la información brindada por Visual Studio usaremos la herramienta opensource “SonarQube”, esta herramienta nos da la posibilidad de realizar análisis de nuestro código a través de una plataforma web, entre las cuales usaremos para eliminar la duplicidad de código y código innecesario.

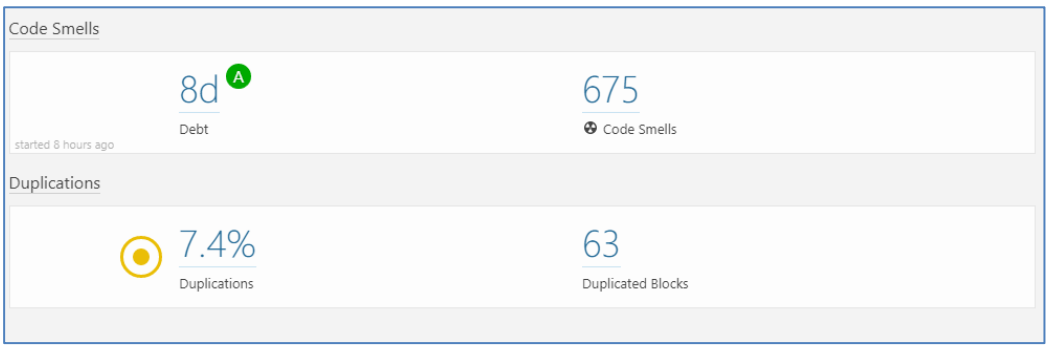


Ilustración 42 - Resultados análisis de nuestro proyecto con SonarQube.

A simple vista podemos notar que el principal problema en el proyecto es la duplicidad de código con un 7.4%. El cual tendremos que analizar a fondo cuando entremos a eliminar líneas de código repetidos.

- ✓ **% de Reutilización de Código:**
  Analizaremos cada formulario, uno por uno, a fin de detectar mismo código que se utilice más de una vez.

- **Login:** En esta pantalla se encontraron 2 formas de validar el usuario y contraseña, el primero es dando clic en el botón iniciar y la segunda forma usando la tecla “Enter” al terminar de ingresar la contraseña.

  Las líneas de código para estar funciones se repiten para ambos casos en su totalidad.

```
private void btnIngresar_Click(object sender, EventArgs e)
{
    string nombre = txtUsuario.Text;
    string password = txtContraseña.Text;

    Usuario u = UsuarioNEG.Instancia().Login(nombre, password);

    if (u.idUsuario > 0)
    {

        //this.Dispose();
        int idUsuario = u.idUsuario;
        String userName = u.Username;
        String nombres = u.Nombres;
        String apellidoPaterno = u.ApellidoPaterno;
        String apellidoMaterno = u.ApellidoMaterno;
        int idTipo = u.idTipoUsuario;

        frmCentral cl = new frmCentral(idUsuario, userName, nombres, apellidoPaterno, apellidoMaterno, idTipo);
        cl.Show();
        this.Hide();

    }
    else { MessageBox.Show("El usuario no esta registrado"); }


}
```

Ilustración 43 - Codificación para el botón "Ingresar"

```
private void txtContraseña_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == Convert.ToChar(Keys.Enter))
    {

        string nombre = txtUsuario.Text;
        string password = txtContraseña.Text;

        Usuario u = UsuarioNEG.Instancia().Login(nombre, password);

        if (u.idUsuario > 0)
        {

            int idUsuario = u.idUsuario;
            String userName = u.Username;
            String nombres = u.Nombres;
            String apellidoPaterno = u.ApellidoPaterno;
            String apellidoMaterno = u.ApellidoMaterno;
            int idTipo = u.idTipoUsuario;

            frmCentral cl = new frmCentral(idUsuario, userName,nombres, apellidoPaterno, apellidoMaterno, idTipo);
            cl.Show();
            this.Hide();

        }
        else { MessageBox.Show("El usuario no esta registrado"); }

    }
```

Ilustración 44 - Codificación para la tecla "Enter"

Para esto caso se implementa la función ***ValidarCredenciales()***, dicha función se utilizará para validar el usuario y contraseña sea cual sea el método de ingreso.

```
public void ValidarCredenciales()
{
    string nombre = txtUsuario.Text;
    string password = txtContraseña.Text;

    Usuario u = UsuarioNEG.Instancia().Login(nombre, password);

    if (u.idUsuario > 0)
    {
        int idUsuario = u.idUsuario;
        String userName = u.Username;
        String nombres = u.Nombres;
        String apellidoPaterno = u.ApellidoPaterno;
        String apellidoMaterno = u.ApellidoMaterno;
        int idTipo = u.idTipoUsuario;

        frmCentral cl = new frmCentral(idUsuario, userName, nombres, apellic
        cl.Show();
        this.Hide();

    }
    else { MessageBox.Show("El usuario no esta registrado"); }
}
```

Ilustración 45 - Función ValidarCredenciales aplicada en formulario Login.

No se encontraron más casos de duplicidad en dicho formulario.

- **Gestión de Clientes:** Líneas de códigos repetidos para la búsqueda, limpieza de formulario, entre otros.

```
private void txtNombre_TextChanged(object sender, EventArgs e)
{
    //ClienteNEG bl = new ClienteNEG();
    Cliente c = new Cliente();
    c.Nombres = txtNombre_Busqueda.Text;
    c.Ruc = txtRuc_Busqueda.Text;
    //dgvClientes.DataSource = bl.Buscar(c);
    dgvClientes.DataSource = ClienteNEG.Instancia().Buscar(c);
}

private void txtRuc_TextChanged(object sender, EventArgs e)
{
    //ClienteNEG bl = new ClienteNEG();
    Cliente c = new Cliente();
    c.Nombres = txtNombre_Busqueda.Text;
    c.Ruc = txtRuc_Busqueda.Text;
    //dgvClientes.DataSource = bl.Buscar(c);
    dgvClientes.DataSource = ClienteNEG.Instancia().Buscar(c);
}
```

Ilustración 46 - Repetición de líneas de código, búsqueda de clientes

A continuación se agregará la función BuscarCliente(), para reemplazar la duplicidad de código para el proceso de búsqueda; que en este caso

se encuentra por un evento en las casillas de texto “al escribir” y en el evento que llama el botón “Consultar”.

```
public void BuscarCliente()
{
    Cliente c = new Cliente();
    c.Nombres = txtNombre_Busqueda.Text;
    c.Ruc = txtRuc_Busqueda.Text;
    dgvClientes.DataSource = ClienteNEG.Instancia().Buscar(c);
}
```

Ilustración 47 - Función Buscar Cliente

Continuando con el análisis, se evidenció nuevamente líneas de código repetidas, en este caso para la función de registro de un nuevo cliente:

```
private void btnGuardar_Click(object sender, EventArgs e)
{
    if (txtId.Text == "")
    {
        Cliente c = new Cliente();
        c.Nombres = txtNombre.Text;
        c.Direccion = txtDireccion.Text;
        c.Ruc = txtRuc.Text;
        c.Telefono = txtTelefono.Text;
        c.Email = txtEmail.Text;

        if (radActivo.Enabled == true)
        {
            c.Estado = true;
        }
        else if (radInactivo.Enabled == true)
        {
            c.Estado = false;
        }
        //ClienteNEG bl = new ClienteNEG();
        // if (bl.Guardar(c) == true)
        if (ClienteNEG.Instancia().Guardar(c) == true)
        {
            MessageBox.Show("El Cliente se grabó correctamente
            this.Dispose();
        }
```

```
else
{
    Cliente c = new Cliente();
    c.idCliente = Convert.ToInt32(txtId.Text);
    c.Nombres = txtNombre.Text;
    c.Direccion = txtDireccion.Text;
    c.Ruc = txtRuc.Text;
    c.Telefono = txtTelefono.Text;
    c.Email = txtEmail.Text;
    if (radActivo.Enabled == true)
    {
        c.Estado = true;
    }
    else if (radInactivo.Enabled == true)
    {
        c.Estado = false;
    }

    //ClienteNEG bl = new ClienteNEG();
    if (ClienteNEG.Instancia().Modificar(c) == true)
    {
        MessageBox.Show("Los datos del Cliente se actualizarón correctamente");
        this.Dispose();
    }
}
```

Ilustración 48 - Lado izquierdo, registro de cliente. Lado derecho, modificación de cliente.

Para este caso bastará con reordenar la codificación aplicando una mejor lógica para el registro o modificación de un Cliente en el sistema, a simple vista podemos notar que para ambos casos se crean una entidad “Cliente”, por otro lado la asignación del Nombre, Dirección, Ruc, Teléfono y Email es la misma para ambos casos; la única variación entre ambos códigos es la verificación de la existencia del código de Usuario, la solución dada fue la siguiente:

```
private void btnGuardar_Click(object sender, EventArgs e)
{
    Cliente c = new Cliente();
    c.Nombres = txtNombre.Text;
    c.Direccion = txtDireccion.Text;
    c.Ruc = txtRuc.Text;
    c.Telefono = txtTelefono.Text;
    c.Email = txtEmail.Text;

    if (radActivo.Enabled == true)
    {
        c.Estado = true;
    }
    else if (radInactivo.Enabled == true)
    {
        c.Estado = false;
    }
    if (txtId.Text == "" && ClienteNEG.Instancia().Guardar(c) == true)
    {
            MessageBox.Show("El Cliente se registró con exito");
            this.Dispose();
    }
    else
    {
        c.idCliente = Convert.ToInt32(txtId.Text);
        if (ClienteNEG.Instancia().Modificar(c) == true)
        {
            MessageBox.Show("Los datos del Cliente se actualizarón correctamente");
            this.Dispose();
        }
```

Ilustración 49 - Solución propuesta para la lógica de registro o modificación de un cliente.

Se unieron ambas líneas de código y se mantiene la funcionalidad inicial aplicada para el registro o modificación de clientes.

No se encontró más duplicidad de código en dicho formulario.

- **Gestión de Artículos:** Para este caso, se están usando las mismas líneas de código para el registro y modificación de un Artículo, la lógica es la misma, no es necesario reescribir nuevamente todo el código.

```
private void btnRegistrar_Click(object sender, EventArgs e)
{
    if (txtIdArticulo.Text == "")
    {
        Articulo a = new Articulo();
        a.Codigo = txtCodigo.Text;
        a.BarCode = txtBarCode.Text;
        a.Nombre = txtNombre.Text;
        a.Descripcion = txtDescripcion.Text;
        a.idColorArticulo = cboColor.SelectedIndex + 1;
        a.idEstiloArticulo = cboEstilo.SelectedIndex + 1;
        a.idGeneroArticulo = cboGenero.SelectedIndex + 1;
        a.idTallaArticulo = cboTalla.SelectedIndex + 1;

        if (radActivo.Checked == true)
        {
            a.Estado = true;
        }
        else if (radInactivo.Checked == true)
        {
            a.Estado = false;
        }
        //ClienteNEG bl = new ClienteNEG();
        // if (bl.Guardar(c) == true)
        if (ArticuloNEG.Instancia().Guardar(a) == true)
        {
            MessageBox.Show("El Articulo se registró correctamente");

        }
```

```
else
{
    Articulo a = new Articulo();
    a.idArticulo = Convert.ToInt32(txtIdArticulo.Text);
    a.Codigo = txtCodigo.Text;
    a.BarCode = txtBarCode.Text;
    a.Nombre = txtNombre.Text;
    a.Descripcion = txtDescripcion.Text;
    a.idColorArticulo = cboColor.SelectedIndex+1;
    a.idEstiloArticulo = cboEstilo.SelectedIndex+1;
    a.idGeneroArticulo = cboGenero.SelectedIndex+1;
    a.idTallaArticulo = cboTalla.SelectedIndex+1;

    if (radActivo.Checked == true)
    {
        a.Estado = true;
    }
    else if (radInactivo.Checked == true)
    {
        a.Estado = false;
    }

    //ClienteNEG bl = new ClienteNEG();
    if (ArticuloNEG._Instancia.Modificar(a) == true)
    {
        MessageBox.Show("El Articulo se modificó correctamente");

    }
}
```

Ilustración 50 - Izquierda: Registro de Articulo, Derecha: Modificar Articulo

Como lo aplicado anteriormente en otro formulario, esto se soluciona reordenando las líneas de códigos, la solución propuesta es la siguiente:

```
private void btnRegistrar_Click(object sender, EventArgs e)
{
    Articulo a = new Articulo();
    a.Codigo = txtCodigo.Text;
    a.BarCode = txtBarCode.Text;
    a.Nombre = txtNombre.Text;
    a.Descripcion = txtDescripcion.Text;
    a.idColorArticulo = cboColor.SelectedIndex + 1;
    a.idEstiloArticulo = cboEstilo.SelectedIndex + 1;
    a.idGeneroArticulo = cboGenero.SelectedIndex + 1;
    a.idTallaArticulo = cboTalla.SelectedIndex + 1;

    if (radActivo.Checked == true)
        {
            a.Estado = true;
        }
    else if (radInactivo.Checked == true)
        {
            a.Estado = false;
        }
    if (txtIdArticulo.Text == "" && ArticuloNEG.Instancia().Guardar(a) == true)
        {
            MessageBox.Show("El Articulo se registró correctamente");
        }
    else
    {
        a.idArticulo = Convert.ToInt32(txtIdArticulo.Text);
    if (ArticuloNEG._Instancia.Modificar(a) == true)
        {
        MessageBox.Show("El Articulo se modificó correctamente");
```

Ilustración 52 - Fragmento de código ordenado para la función Registro de Artículos.

Similar a lo aplicado anteriormente en el formulario de Gestión de Clientes, nuevamente reducimos en un 50% la cantidad de código aplicada anteriormente en la lógica de registro y modificación.

Ilustración 51 - Registro y modificación de Artículos

- **Gestión de Colores de Artículos:** En este formulario solo se realizan las funciones de registro y modificación, dado que en todos los formularios de proyecto se usó la misma lógica, se aplicara la misma propuesta realizada anteriormente para los formularios de Gestión de Artículos y Gestión de Clientes.

```
private void btnGuardar_Click(object sender, EventArgs e)
{
    if (txtId.Text == "")
    {
        ColorArticulo c = new ColorArticulo();
        c.Descripcion = txtDescripcionColor.Text;

        if (radActivo.Enabled == true)
        {
            c.Estado = true;
        }
        else if (radInactivo.Enabled == true)
        {
            c.Estado = false;
        }

        if (ColorArticuloNEG.Instancia().Guardar(c) == true)
        {
            MessageBox.Show("El Color se grabó correctamente");
            DialogResult result2 = MessageBox.Show("Desea registrar otro Color?", "Cerrar", Messag
            if (result2 == DialogResult.Yes)
            {
                txtDescripcionColor.Clear();
            }
            else if (result2 == DialogResult.No)
            {
                this.Dispose();
            }
        }
    }
```

Ilustración 53 - Lógica de registro y modificación de Colores

```
private void btnGuardar_Click(object sender, EventArgs e)
{
    ColorArticulo c = new ColorArticulo();
    c.Descripcion = txtDescripcionColor.Text;

        if (radActivo.Enabled == true)
        {
            c.Estado = true;
        }
        else if (radInactivo.Enabled == true)
        {
            c.Estado = false;
        }

    if (txtId.Text == "" && ColorArticuloNEG.Instancia().Guardar(c) == true)
        {
            MessageBox.Show("El Color se grabó correctamente");
            DialogResult result2 = MessageBox.Show("Desea registrar otro Color?", "Cerrar", MessageBoxButt
            if (result2 == DialogResult.Yes)
            {
                txtDescripcionColor.Clear();
            }
            else if (result2 == DialogResult.No)
            {
                this.Dispose();
            }
        }
    else
    {
        c.idColor = Convert.ToInt32(txtId.Text);
        if (ColorArticuloNEG.Instancia().Modificar(c) == true)
```

Ilustración 54 - Lógica Propuesta de Registro y Modificación de Colores

El resto de pantallas también contienen estos problemas, no será necesario indicar todos en el informe, ya que se verá reflejado en los resultados. Con el uso de funciones corregiríamos todos estos problemas, así mismo mejorará el rendimiento del software.

- ✓ **Eliminación de código innecesario:**

Ahora pasamos a la eliminación de código innecesario, básicamente tenemos 3 tipos de código: código muerto, código redundante y código inalcanzable.

Hay que tener en cuenta que la mayor parte del código de una aplicación se concentra en la "Capa de Presentación", esto lo podemos corroborar con el análisis de métricas realizado anteriormente.

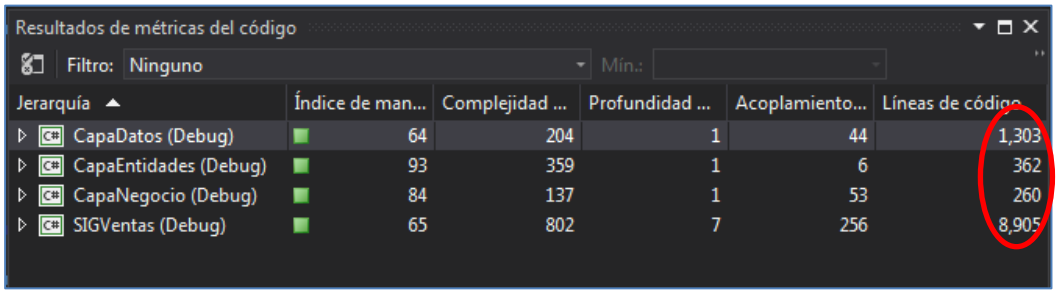


Ilustración 55 - Comparación Líneas de Código

En la mayoría de formularios encontramos casos de código muerto, a continuación se presentan un par de imágenes con eventos en blanco.

```
private void Login_Load(object sender, EventArgs e)
{

}
```

Ilustración 56 - Evento Load en blanco

```
private void txtRuc_TextChanged_1(object sender, EventArgs e)
{


}
```

Ilustración 57 - Evento en campo de texto en blanco

También se observó en algunos formularios validaciones mal codificadas como la siguiente:

```
private void btnRegistrarVenta_Click(object sender, EventArgs e)
{

    if (this.codigoClienteSeleccionado == -1)
    {
        this.mError("No ha seleccionado aún ningun Cliente");
    }

    if (txtCodigo.Text =="")
    {
        String NroActual;
        Venta nroVentaActual = VentaNEG.Instancia().obtenerUltimaVenta(cboTipo

        if (nroVentaActual.NroDocumento == "")
        {
```

Ilustración 58 - Validación no formulada de manera correcta

```
private void nudCantidad_ValueChanged(object sender, EventArgs e)
{
    //if (nudCantidad.Value.ToString() ;
}
```

Ilustración 59 - Evento que nunca es llamado en el formulario.

```
private void button3_Click(object sender, EventArgs e)
{
    PrinterSettings settings = new PrinterSettings();
    MessageBox.Show(settings.PrinterName, "Cerrar", MessageBoxButtons.YesNo);


    string printerName = settings.PrinterName;
    string query = string.Format("SELECT * from Win32_Printer WHERE Name LIKE '%{0}'", printerName);
    ManagementObjectSearcher searcher = new ManagementObjectSearcher(query);
    ManagementObjectCollection coll = searcher.Get();

    foreach (ManagementObject printer in coll)
    {
        foreach (PropertyData property in printer.Properties)
        {
            MessageBox.Show(string.Format("{0}: {1}", property.Name, property.Value), "Cerrar", MessageBoxButtons.
            Console.WriteLine(string.Format("{0}: {1}", property.Name, property.Value));
        }
    }
}
```

Ilustración 60 - Evento de Botón que nunca se llama, parece ser código de prueba que no fue borrado

```
private void radActivo_CheckedChanged(object sender, EventArgs e)
{

}

private void radInactivo_CheckedChanged(object sender, EventArgs e)
{

}
```

Ilustración 61 - Nuevamente eventos que no se usan en ningún momento.

Para varios de los formularios la función “IF” siempre está acompañada de un “IF ELSE” no siendo siempre necesario. (Ver ilustración 63)

```
if (radActivo.Enabled == true)
{
    c.Estado = true;
}
else if(radInactivo.Enabled == true)
{
    c.Estado = false;
}
```

Ilustración 62 - Uso de IF ELSE de manera innecesaria.

Para esta verificación solo tenemos 2 condiciones, no es necesario realizar todas las comparaciones, ya que si no es la primera, únicamente puede ser la segunda opción.

Asimismo en la misma ilustración podemos notar que se agrega la línea de código “==true” en la condición “if”, hay que tener en cuenta que no es necesario agregar ese segmento de código si se trata de una afirmación, en este caso bastara solo las siguientes líneas de código:

```
if (radActivo.Enabled)
{
    c.Estado = true;
}
else
{
    c.Estado = false;
}
```

Ilustración 63 - Nueva codificación para la selección del estado.

Son muchas las observaciones que se pueden encontrar en los formularios, la solución es sencilla: eliminar todos esos fragmentos de código sueltos y sin uso.

Para realizar una mejor identificación de la mayoría de segmentos de código innecesario nos apoyaremos en la aplicación SonarQube:

	Duplicated Lines (%)	Duplicated Lines
CapaDatos	9.6%	290
SIGVentas	7.8%	1,339
CapaNegocio	0.0%	0
CapaEntidades	0.0%	0

Ilustración 64 - Resultados análisis de código repetido en la aplicación.

La aplicación nos muestra por cada clase y formulario las porciones de código duplicado, y nos da un porcentaje del total.

	Duplicated Lines (%)	Duplicated Lines
Sunat.cs	87.9%	210
Sunarp.cs	87.9%	210
frmReporteVentas.cs	63.8%	104
frmReporteInventarioAlmacen.cs	58.9%	93
frmGestionServicios.cs	43.4%	109
frmGestionPersonal.cs	35.3%	96
Login.cs	29.4%	20
frmGestionVentas.cs	27.4%	169
frmAnularVentaVerificar.cs	19.8%	23
frmGestionColores.cs	18.6%	30
frmGestionTallas.cs	18.5%	36
frmGestionEstilos.cs	18.5%	36
frmGestionStock.cs	16.2%	48
frmGestionIngresos.cs	15.6%	50

Ilustración 65 - Listado de clases indicando líneas de código duplicadas.

Tomaré como ejemplo el primer archivo “Sunat.cs”, en donde está la codificación para poder obtener la información de un RUC. No nos centraremos en detallar las observaciones de cada clase disponible en el proyecto.

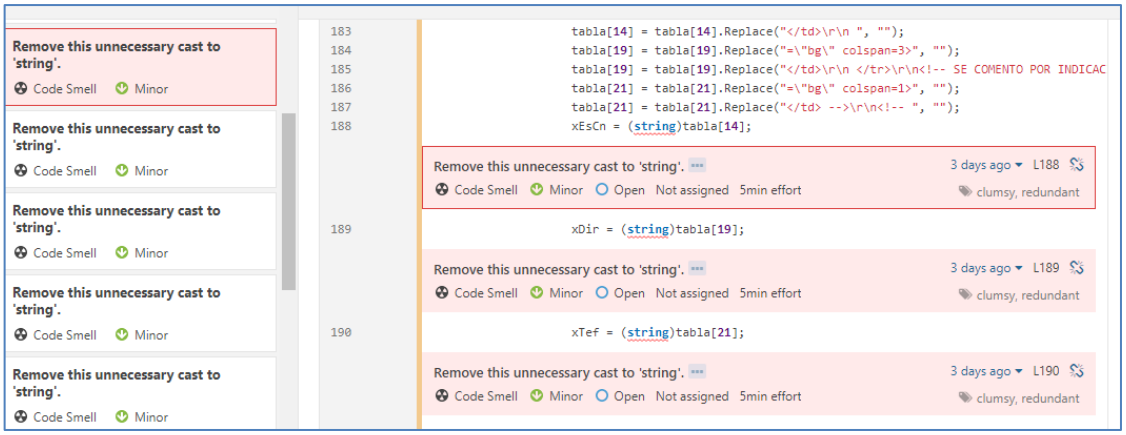


Ilustración 66 - Observaciones encontradas en la clase "Sunat.cs".

La plataforma nos indica, según el impacto en la aplicación, líneas de código innecesario y repetido, en este caso la línea “(string)” que de manera innecesaria se está repitiendo a pesar que la serie de caracteres “tabla[]” ya se había definido como tal.

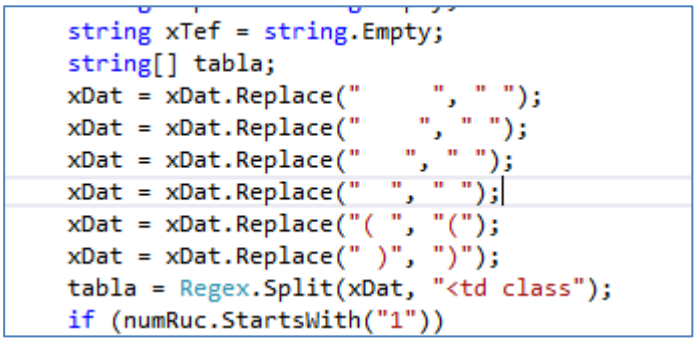


Ilustración 67 - Definición de "string[] tabla".

En este caso la solución es eliminar el fragmento: “(string)” de las líneas de código afectadas.

- ✓ **Validación de la información INPUT:**

  Acá encontré bastantes inconsistencias, se adjuntan imágenes con varios casos de falta de validación en los INPUT.

  En los formularios de registro de Clientes y Personal encontramos:

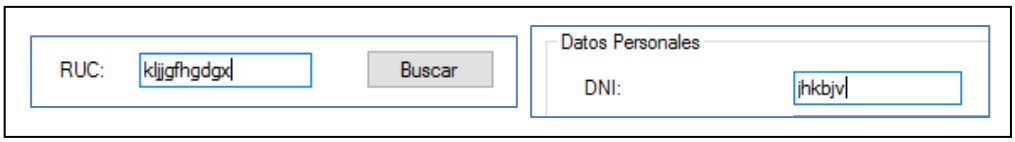


Ilustración 68 – Campos RUC y DNI que permiten ingresar letras.

Para todos los formularios que presenten un problema similar a la mostrada en la Ilustración 60, hay que tener en cuenta que en nuestro país el RUC y el DNI únicamente está conformado por números, usaremos la función:

```
public void IngresoNumeros(KeyPressEventArgs e)
{
    if (Char.IsNumber(e.KeyChar)) //Al pulsar teclas de números.
    {
        e.Handled = false; //Se acepta
    }
    else if (Char.IsControl(e.KeyChar)) //Al pulsar teclas como Borrar y eso.
    {
        e.Handled = false; //Se acepta
    }
    else
    {
        e.Handled = true; //No se acepta
    }
}
```

Ilustración 69 – Función validación para ingreso de solo números.

Con dicho fragmento de código, no solo permitimos el ingreso de sólo números en los DNI, sino también usaremos la misma función de validar el ingreso del RUC. Con la creación de la función evitamos tener que repetir el código completo en cada evento.

```
private void txtRuc_KeyPress(object sender, KeyPressEventArgs e)
{
    IngresoNumeros(e);
}

private void txtRuc_Busqueda_KeyPress(object sender, KeyPressEventArgs e)
{
    IngresoNumeros(e);
}
```

Ilustración 70 - Validación para le ingreso de solo números en el campo RUC

En el formulario de registro de ventas de igual manera no existe la validación para el ingreso de datos.

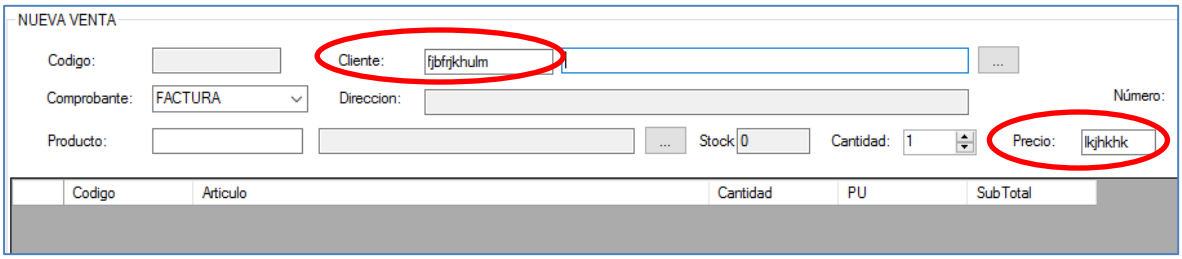


Ilustración 71 - Campos Cliente y Precio permiten el ingreso de caracteres erróneos.

Inclusive el ComboBox Comprobante está aceptando el ingreso de valores, únicamente cuando se debe seleccionar un dato específico, en este caso: Factura y Boleta.

Para solucionar este problema, únicamente tenemos que cambiar la propiedad **DropDownStyle,** por defecto al agregar el control la configurar del DropDownStyle es:

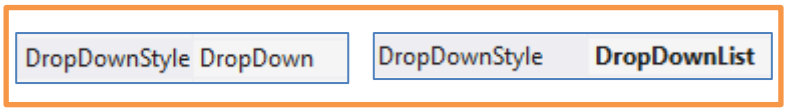


Ilustración 72 – Izquierda: Configuración por defecto, derecha configuración utilizada para bloquear la edición.

Otro problema detectado en todos los formularios es que se permite el registro de Clientes, Artículos, Ingreso, etc a pesar de que no se halla ingresado dato algunos en los controles de textos.

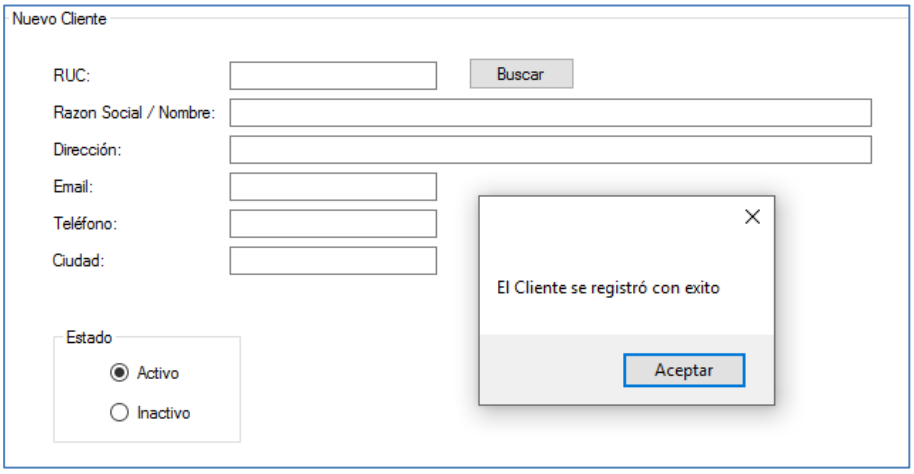


Ilustración 73 - Registro de Cliente sin datos

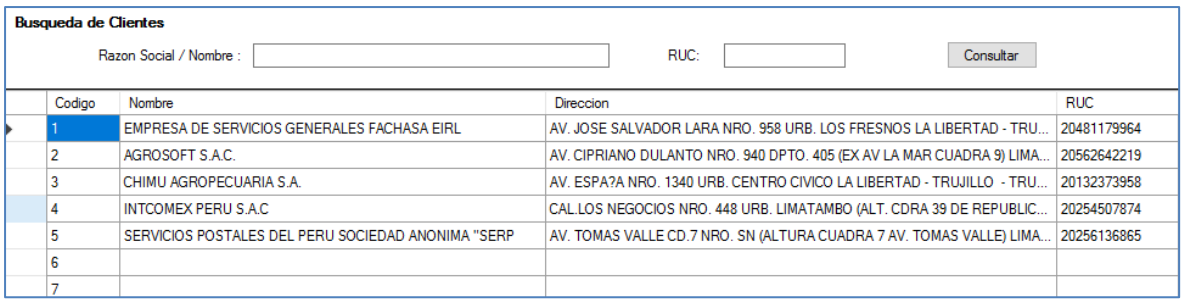


Ilustración 74 - Prueba de registro en blanco.

Esto se debe a que al momento de dar clic en el botón guardar no se verifica si todos los campos que sean obligatorios estén llenos. La siguiente función solucionara este problema:

```
private bool ValidarDatos()
{
    foreach (Control c in this.gboxRegistro.Controls)
    {
        if (c is TextBox)
        {
            if (string.IsNullOrWhiteSpace(((TextBox)c).Text))
            {
                MessageBox.Show("No pueden quedar espacios en blanco.");
                return false;
            }
        }
    }
    return true;
}
```

Ilustración 75 - Función para validar que todos los campos tengan datos.

Este fragmento de código evaluará si todos los “TextBox” del GroupBox, en este caso, “gboxRegistro” no son nulos ni están en blanco, de ser verdadero no nos permitirá guardar.

```
private void btnGuardar_Click(object sender, EventArgs e)
{

    if (ValidarDatos())
    {

        email_bien_escrito(txtEmail.Text);
        Cliente c = new Cliente();
        c.Nombres = txtNombre.Text;
        c.Direccion = txtDireccion.Text;
        c.Ruc = txtRuc.Text;
        c.Telefono = txtTelefono.Text;
        c.Email = txtEmail.Text;

        if (radActivo.Enabled == true)
        {
            c.Estado = true;
        }
        else if (radInactivo.Enabled == true)
```

Ilustración 76 - Llamado de la función "ValidarDatos" al momento de guardar.

Esta función se replicará para todos los formularios que tengan la función de guardar, esto solo sirve si se quiere validar que todos los campos sin excepción tengan datos.

Algo que sobresalió al momento de realizar la validación en el registro de datos de los formularios es que todos los mensajes emitidos en "MessageBox" poseen únicamente un mensaje de texto sin ningún tipo de icono ni título, considero que no es de mucha ayuda ya que no indicará si el usuario está cometiendo algún tipo de error en el registro.

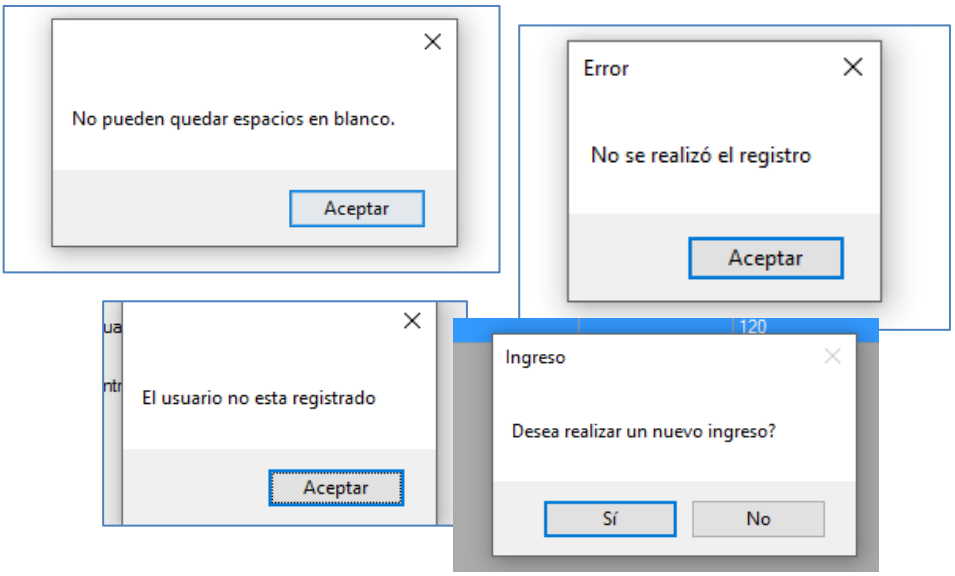


Ilustración 77 - Ejemplos de mensajes de información.

Se agregará los iconos correspondientes a cada mensaje, así mismo los títulos de estos, ya que en algunos casos no se tiene ningún título.

Actualmente las líneas de código usadas para la creación de estos mensajes es la siguiente:

MessageBox.Show("No se realizó el registro", "Info");

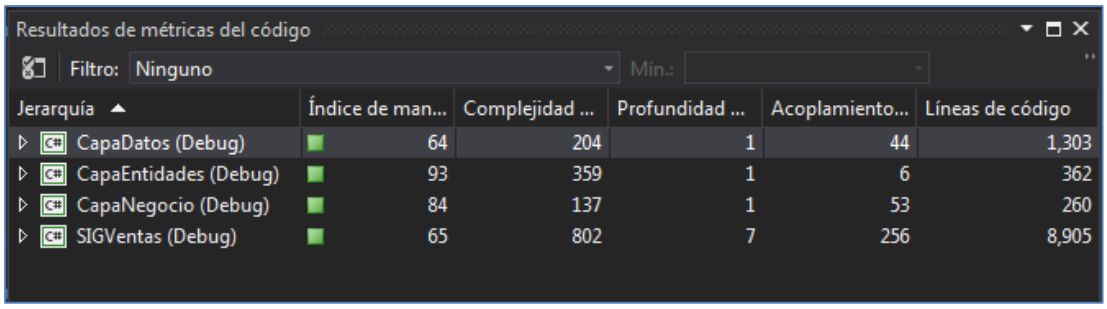
Lo ideal es la siguiente:

```
MessageBox.Show("No se realizó el registro", "Aviso",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
```

4. RESULTADOS

Después de realizar todos los cambios en la codificación del proyecto obtenemos los siguientes resultados:

Resultados métricas Visual Studio 2015:



Jerarquía	Índice de man...	Complejidad ...	Profundidad ...	Acoplamiento...	Líneas de código
CapaDatos (Debug)	64	204	1	44	1,303
CapaEntidades (Debug)	93	359	1	6	362
CapaNegocio (Debug)	84	137	1	53	260
SIGVentas (Debug)	65	802	7	256	8,905

Ilustración 78 - Resultados métricas antes de los cambios.

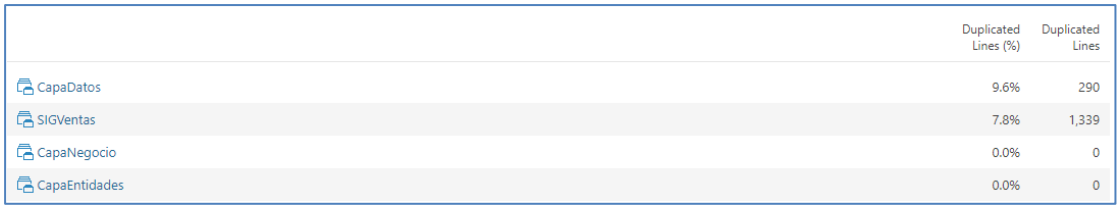
Jerarquía	Índice de man...	Complejidad c...	Profundid...	Acoplamiento ...	Líneas de código
SIGVentas (Debug)	69	728	7	266	8,646
CapaEntidades (Debug)	93	359	1	6	362
CapaNegocio (Debug)	84	137	1	53	260
CapaDatos (Debug)	64	204	1	43	1,302

Ilustración 79 - Resultados después de los cambios.

Estos resultados aplican en cuanto a la eliminación de código duplicado e innecesario, de los cuales podemos deducirlo siguiente:

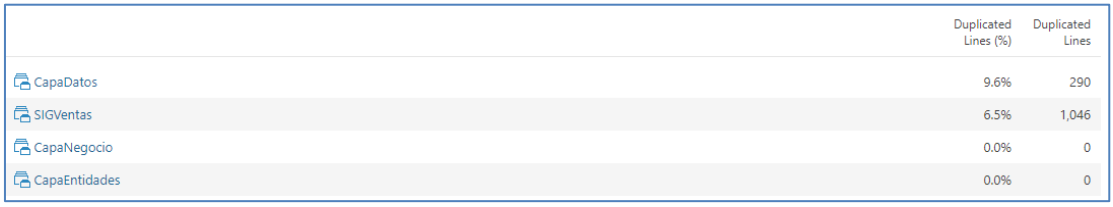
- Aumento la mantenibilidad del proyecto de 64 a 69.
- La complejidad ciclomatica se redujo de 802 a 728
- Las líneas de código se redujeron de 8905 a 8646.

Por otro lado, en el nuevo análisis realizo en SonarQube obtuvimos los siguientes resultados:



	Duplicated Lines (%)	Duplicated Lines
CapaDatos	9.6%	290
SIGVentas	7.8%	1,339
CapaNegocio	0.0%	0
CapaEntidades	0.0%	0

Ilustración 80 - Resultados iniciales en la plataforma SonarQube



	Duplicated Lines (%)	Duplicated Lines
CapaDatos	9.6%	290
SIGVentas	6.5%	1,046
CapaNegocio	0.0%	0
CapaEntidades	0.0%	0

Ilustración 81 - Nuevos resultados en la plataforma SonarQube

Con lo que se deduce lo siguiente:

- Las líneas de código repetidas se redujeron de un 7.8% a 6.5%

Como dato adicional la plataforma SonarQube nos brinda una evaluación exitosa para nuestra aplicación, en este caso está identificada con el término “Passed”.

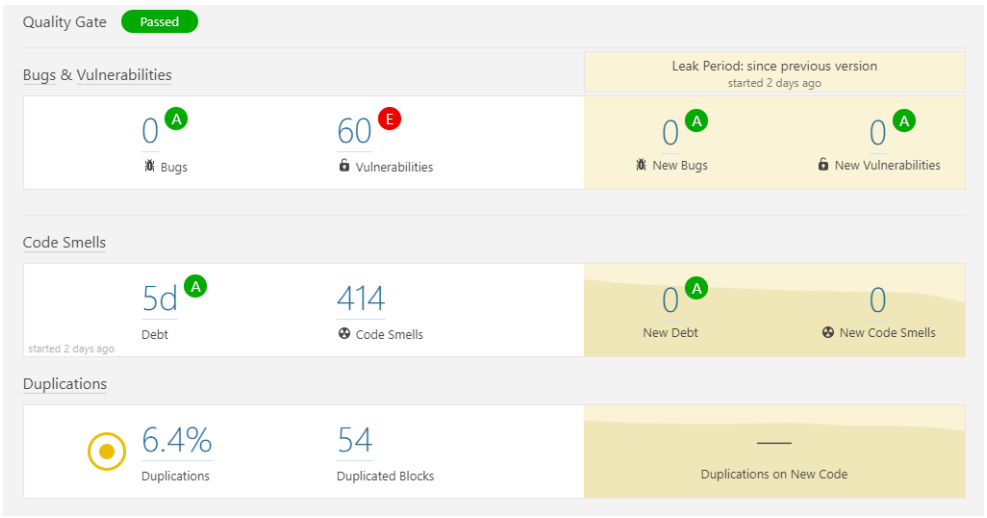


Ilustración 82 - Calificación otorgada por SonarQube.

Se eliminaron el 100% de las faltas de validación encontradas, sin la necesidad de aumentar código en el aplicativo, esto se demuestra en el análisis inicial emitido por VisualStudio y Sonar Qube.

Por otro lado para la refactorización del código del software se utilizaron los siguientes tipos:

- Consolidación de Fragmentos de Condicionales Duplicados: Se encontró gran cantidad de condicionales duplicados y codificados de manera incorrecto.
- Descomposición de condicionales: Se mejoró la codificación de los condicionales codificados de una forma poco práctica.
- Extracción de Métodos: Eliminar duplicidad de código encontrado en todo el código del software.
- Renombre de los Métodos: Los métodos fueron renombrados para ayudar en la mantenibilidad del software.
- Eliminación de Parámetros: Se eliminó los parámetros que no tenían uso alguno.
- Reemplazo de Códigos de Error con Excepciones: Los mensajes de error que no tienen significado alguno para los usuarios finales fueron reemplazados.

5. CONCLUSIONES

- El rediseño de la capa de aplicación del aplicativo de software garantizará una mejor usabilidad, la interfaz es más limpia y clara para el usuario, siendo fácil te entender luego de mejorar los mensajes emitidos.
- El software no tenía implementado Tooltips, que eran sumamente necesarios especialmente en los formularios con gran cantidad de controles que podrían confundir al usuario.
- SonarQube es una herramienta interesante que nos da varias características que nos puede ofrecer un software libre, en este proyecto solo se usó para la eliminación de código repetido e innecesario.
- Las líneas de comentarios no son necesarias si el programa está bien codificado y claro para una posterior mejora por parte de otros programadores.

6. RECOMENDACIONES

- Se recomienda tener la documentación adecuada para el software, actualmente el software no cuenta con ningún tipo de documentación.
- El software no cuenta con un formulario de ayuda (“Manual de usuario”), es recomendable al menos tenerlo de manera física con el fin de mejorar el desempeño de los usuarios ante un posible problema en el uso del aplicativo.

7. REFERENCIAS BIBLIOGRAFICAS

- Alejandro Floría (2000). Prototipado Rápido. Universidad de Zaragoza, España. Recuperado el 20 de Octubre, de Sidar.org: https://www.sidar.org/recur/desdi/traduc/es/visitable/nuevos/Rapido.htm

- Bass Len, Clements Paul & Kazman Rick. (2012). *Software Architecture in Practice*. San Francisco: Addison-Wesley.

- Cesar de La Torre, Unai Zorrilla, Miguel Angel & Javier Calvarro. (2010). *Guia de Arquitectura N Capas orientada al Dominio con .NET 4.0.* Recuperado el 10 de diciembre, de WPF: http://wpf.com.es/wp-content/uploads/2014/02/Guia_Arquitectura_N-Capas_DDD_NET_4_Borrador_Marzo_2010.pdf

- IEEE Computer Society. (2004). *Swebok, un proyecto de la práctica profesional del IIEE Computer Society*. California.

- Jakob Nielsen. (2012). Usability 101: Introduction to Usability. Recuperado el 13 de Noviembre, de Nielsen Norman Group: https://www.nngroup.com/articles/usability-101-introduction-to-usability/

- Juan Martínez. (1999) *Estándar IEEE 1219 de mantenimiento del Software.* (Proyecto final de carrera). Universidad de Castilla, España.

- L.J. Arthur. (1988) *Software Evolution: The Software Maintenance Challenge*. John Wiley & Sons, Nueva York.

- Mario Lorenzo Alcala. (2006). *Medida de Usabilidad en Aplicaciones de Escritorio. Un método Práctico*. (Proyecto final de carrera). Universidad Nacional de Educación a Distancia, España.

- Maximiliano Mascheroni, Cristina Greiner & Raquela Pretis. (2013). *Técnicas de Usabilidad. Estudio Exploratorio sobre su incorporación en los procesos de desarrollo de software en Pymes Locales.* Recuperado

el 5 de Enero, de Universidad Tecnológica Nacional: http://www.frre.utn.edu.ar/IIJCyT/clean/files/get/item/2196

- Microsoft Patterns & Practices Team. (2009) *Microsoft Application Architecture Guide*. Recuperado el 12 de diciembre del 2016, de Microsoft: https://msdn.microsoft.com/en-us/library/ee658081.aspx

- M. Dorfman and R. H. Thayer. (1997) *Software Engineering*. Estados Unidos: IEEE Computer Society Press.

- Organización Internacional de Normalización. (2002) *Estándar ISO/IEC TR 9126-3*. Recuperado el 10 de enero, de Computer Science and Engineering: www.cse.unsw.edu.au/~cs3710/PMmaterials/Resources/9126-3%20Standard.doc

- Roger S. Pressman. (2010). *Ingeniería del Software, un Enfoque Práctico*. México: Mc. Graw Hill.

- R.S. Arnold. (1992) *Software Reengineering*. Estados Unidos: IEEE Computer Society.

- Unidad de Modernización y Gobierno Digital (2008). ¿Qué es la Usabilidad?, Ministerio Secretaría General de la Presidencia, Chile. Recuperado el 13 de Noviembre, de GuiaDigital.gob.cl: http://www.guiadigital.gob.cl/articulo/que-es-la-usabilidad

- Walter Sanchez (2011). La usabilidad en Ingeniería de Software: definición y características. Universidad Don Bosco, El Salvador.

- Xavier Ferré Grau. (2011). *Principios Básicos de Usabilidad para Ingenieros Software*. Boadilla del Monte, Madrid.

- Martin Fowler (2013). Catalog of Refactorings. Recuperado el 13 de Marzo, de Refactoring.com: https://refactoring.com/catalog

APENDICE 1: Resultado Casos de Prueba

Id	Caso de Prueba	Descripción	Fecha	Funcionalidad / Característica	Datos / Acciones de Entrada	Resultado Esperado	**Resultado Obtenido**	**Estado**	**Observaciones**
1	Gestión Clientes	Registro, Modificación, búsqueda clientes	12/12/2016	Registro Nuevo	DNI: DF73FHCJS - Guardar	Mensaje indicando que el RUC no es el correcto, no debe permitir registro	No exige corrección y permite el registro.	Incorrecto	El RUC debe ser números y 11 dígitos.
2	Gestión Clientes	Registro, Modificación, búsqueda clientes	12/12/2016	Registro Nuevo	Campos sin datos - Guardar	Mensaje indicando que los campos deben ser llenados, no debe permitir registro	No exige que los campos estén con datos y permite el registro.	Incorrecto	No aplica validación para ningún campo requerido, no todos los campos deben ser obligatorios.
3	Gestión Clientes	Registro, Modificación, búsqueda clientes	12/12/2016	Registro Nuevo	RUC ya registrado - Guardar	Mensaje indicando que el RUC ya está registrado, no debe permitir registro	La pantalla cambia a modificación de registro, no permite registro doble.	Correcto	
4	Gestión Personal	Registro, Modificación, búsqueda Personal	12/12/2016	Registro Nuevo	DNI ya registrado - Guardar	Mensaje indicando que el DNI ya está registrado, no debe permitir registro	Se completa el registro con éxito.	Incorrecto	No debe haber duplicidad de DNI en el sistema.
5	Gestión Personal	Registro, Modificación, búsqueda Personal	12/12/2016	Registro Nuevo	Salario ingresado letras - Guardar	No permitir el ingreso de caracteres que no son números , no debe permitir registro	Se completa el registro con éxito.	Incorrecto	Se debe validar el campo para evitar el ingreso de otros caracteres que no sean números.

APENDICE 2: Diccionario de Datos

CURN-001

Características de la Tabla			
Nombre de la Tabla	Cliente		
Descripción de la Tabla	Esta tabla fue diseñada para permitir el almacenamiento de los Clientes.		
Campos de la Tabla			
Nombre del Campo	**Tipo de Dato**	**Longitud del Campo**	**Breve Descripción del Campo**
idCliente	varchar	20	Identificador único del cliente.
Nombres	varchar	100	Es el campo donde se almacenan los nombres del cliente.
Dirección	int	1	Campo donde se guarda la dirección del cliente.
Email	varchar	50	Es el campo donde se almacenan la cuenta de correo del cliente.
Ruc	varchar	50	Es el campo donde se almacena el RUC del cliente
Estado	bit	1	Estado del color, nos indica si esta activa o inactivo.

CURN-002

Características de la Tabla			
Nombre de la Tabla	Articulo		
Descripción de la Tabla	Esta tabla fue diseñada para permitir el almacenamiento de registro de los diferentes artículos		
Campos de la Tabla			
Nombre del Campo	**Tipo de Dato**	**Longitud del Campo**	**Breve Descripción del Campo**
BarCode	varchar	50	Código de barras del articulo
Codigo	varchar	50	Código identificación del artículo.
Descripcion	varchar	100	Descripción larga del artículo.
idArticulo	int	1	Es el identificador único del artículo.
idColorArticulo	int	1	Campo donde se guarda el tipo de color del artículo. Relación con ColorArticulo
idEstiloArticulo	Int	1	Campo donde se guarda el estilo del artículo. Relación con EstiloArticulo
Nombre	varchar	50	Nombre del artículo.

CURN-003			
Características de la Tabla			
Nombre de la Tabla	ColorArticulo		
Descripción de la Tabla	Esta tabla fue diseñada para permitir el almacenamiento de los diferentes tipos de color para los artículos.		
Campos de la Tabla			
Nombre del Campo	**Tipo de Dato**	**Longitud del Campo**	**Breve Descripción del Campo**
idColor	int	11	Identificador único del color.
Nombre	varchar	50	Es el campo donde se almacena la descripción del tipo de cuenta
Estado	bit	1	Campo donde se almacena 1 o 0 el estado del Color.

CURN-004			
Características de la Tabla			
Nombre de la Tabla	EstiloArticulo		
Descripción de la Tabla	Esta tabla fue diseñada para permitir el almacenamiento de los diferentes tipos de color para los artículos.		
Campos de la Tabla			
Nombre del Campo	**Tipo de Dato**	**Longitud del Campo**	**Breve Descripción del Campo**
idEstilo	int	11	Identificador único del estilo.
Nombre	varchar	50	Es el campo donde se almacena la descripción del tipo de cuenta
Estado	bit	1	Campo donde se almacena 1 o 0 el estado del Estilo.

CURN-005			
Características de la Tabla			
Nombre de la Tabla	GeneroArticulo		
Descripción de la Tabla	Esta tabla fue diseñada para permitir el almacenamiento de los diferentes tipos de color para los artículos.		
Campos de la Tabla			
Nombre del Campo	**Tipo de Dato**	**Longitud del Campo**	**Breve Descripción del Campo**
idGenero	int	11	Identificador único del género.
Nombre	varchar	50	Es el campo donde se almacena la descripción del tipo de cuenta
Estado	int	1	Campo donde se almacena 1 o 0 el estado del Genero.

CURN-006

Características de la Tabla			
Nombre de la Tabla	DetalleVenta		
Descripción de la Tabla	Esta tabla fue diseñada para el almacenamiento del detalle de las ventas.		
Campos de la Tabla			
Nombre del Campo	**Tipo de Dato**	**Longitud del Campo**	**Breve Descripción del Campo**
idVenta	int	1	Identificador único de la venta a la que pertenece el DetalleVenta.
idDetalleVenta	int	1	Identificador único del detalle de la venta.
idDetalleIngresoAlmacen	int	1	Identificador único del ingreso de almacén que pertenece al DetalleVenta.
Cantidad	int	50	Campo con la cantidad de productos vendidos.
idArticulo	Int	1	Identificador del Artículo vendido.
PrecioVenta	decimal	18,2	Precio de venta del Artículo vendido.

CURN-007

Características de la Tabla			
Nombre de la Tabla	Venta		
Descripción de la Tabla	Esta tabla fue diseñada para el almacenamiento de las ventas.		
Campos de la Tabla			
Nombre del Campo	**Tipo de Dato**	**Longitud del Campo**	**Breve Descripción del Campo**
idVenta	int	1	Identificador único de la Venta.
idCliente	int	1	Identificador único del Cliente al cual se le realizó la venta
idUsuario	int	1	Identificador único del usuario que realizó la venta.
idTipoComprobante	int	50	Identificador único del tipo de comprobante de la venta.
Correlativo	varchar	50	Número Correlativo de la venta.
NroDocumento	varchar	50	Número de documento de la venta.

FechaVenta	datetime		Fecha de la Venta
Estado	bit	1	Campo con el estado de la venta.
TotalVenta	Decimal	18,2	Monto total de la venta.

CURN-008

Características de la Tabla			
Nombre de la Tabla	Usuario		
Descripción de la Tabla	Esta tabla fue diseñada para el almacenamiento de los usuarios que usaran el sistema.		
Campos de la Tabla			
Nombre del Campo	**Tipo de Dato**	**Longitud del Campo**	**Breve Descripción del Campo**
ApellidoPaterno	int	1	Campo con el apellido paterno del usuario.
ApellidoMaterno	int	1	Campo con el apellido materno del usuario.
Nombres	int	1	Campo con los nombres del usuario.
Direccion	varchar	50	Campo con la dirección del usuario.
Dni	varchar	50	Campo con el DNI del usuario.
Estado	bit	1	Campo que indicado si el usuario está activo o inactivo.
FechaIngreso	datetime		Fecha de Ingreso a almacén.
idTipoUsuario	int	1	Identificador del Tipo de Usuario.
idUsuario	Decimal	18,2	Identificador Unido del usuario
Username	varchar	50	Campo con el nombre de usuario para acceder al sistema.
Password	varchar	50	Campo con la contraseña del usuario para acceder al sistema.

CURN-009

Características de la Tabla	
Nombre de la Tabla	TipoUsuario

Descripción de la Tabla	Esta tabla fue diseñada para el almacenamiento de los tipos de usuarios		
Campos de la Tabla			
Nombre del Campo	**Tipo de Dato**	**Longitud del Campo**	**Breve Descripción del Campo**
idTipoUsuario	int	1	Identificador único del tipo de usuario.
Descripcion	varchar	50	Campo con la descripción del tipo de Usuario.

CURN-010

Características de la Tabla			
Nombre de la Tabla	IngresoAlmacen		
Descripción de la Tabla	Esta tabla fue diseñada para el almacenamiento de los ingresos de productos al Almacén.		
Campos de la Tabla			
Nombre del Campo	**Tipo de Dato**	**Longitud del Campo**	**Breve Descripción del Campo**
idIngresoAlmacen	int	1	Identificador único del ingreso a almacén.
idProveedor	int	1	Identificador del Proveedor al que se realiza la compra de productos.
idTipoComprobante	int	1	Identificador del tipo de Comprobante.
FechaIngreso	datetime		Campo con la fecha de ingreso a almacén.
Estado	bit	1	Campo con el estado activo o inactivo del Ingreso.
Correlativo	varchar	50	Campo con el correlativo del comprobante.
idUsuario	int	1	Identificador del usuario que realizo el ingreso.
TotalPagado	decimal	10,2	Campo con el monto total pagado al proveedor.

CURN-011

Características de la Tabla			
Nombre de la Tabla	DetalleIngresoAlmacen		
Descripción de la Tabla	Esta tabla fue diseñada para el almacenamiento de los ingresos de productos al Almacén.		
Campos de la Tabla			
Nombre del Campo	**Tipo de Dato**	**Longitud del Campo**	**Breve Descripción del Campo**
idDetalleIngresoAlmacen	int	1	Identificador único del detalle de Ingreso de Almacén.

idArticulo	int	1	Identificador del Artículo el cual ingresa al Almacén.
idIngresoAlmacen	int	1	Identificador de Ingreso almacén
PrecioCompra	decimal	10,2	Campo con el precio de compra del artículo.
PrecioVenta	decimal	10,2	Campo con el precio de venta del artículo.
Cantidad	int	50	Campo con la cantidad de artículos que ingresan a almacén.
CantidadFinal	int	1	Campo con nueva cantidad de artículos si ya existían en stock.

CURN-012

Características de la Tabla			
Nombre de la Tabla	StockTienda		
Descripción de la Tabla	Esta tabla fue diseñada para el almacenamiento del stock disponible en la tienda.		
Campos de la Tabla			
Nombre del Campo	**Tipo de Dato**	**Longitud del Campo**	**Breve Descripción del Campo**
idStockTienda	int	1	Identificador único del stock en la tienda
idUsuario	int	1	Identificador del Usuario que hace el ingreso de stock
Estado	bit	1	Estado del ingreso de stock.
FechaIngresoTienda	datetime		Fecha de registro de ingreso de stock a tienda.

CURN-013

Características de la Tabla			
Nombre de la Tabla	DetalleStockTienda		
Descripción de la Tabla	Esta tabla fue diseñada para el almacenamiento del stock disponible en la tienda.		
Campos de la Tabla			
Nombre del Campo	**Tipo de Dato**	**Longitud del Campo**	**Breve Descripción del Campo**
idStockTienda	int	1	Identificador único del stock en la tienda
idDetalleStockTienda	int	1	Identificador único del Detalle de Stock de Tienda.

idArticulo	int	1	Identificador del articulo ingresado en el detalle de stock de tienda.
idDetalleIngresoAlmacen	int	1	Identificador Detalle Ingreso almacén.
Stock Final	int	4	Campo con cantidad stock final.
Stock Inicial	int	4	Campo con cantidad de stock inicial.