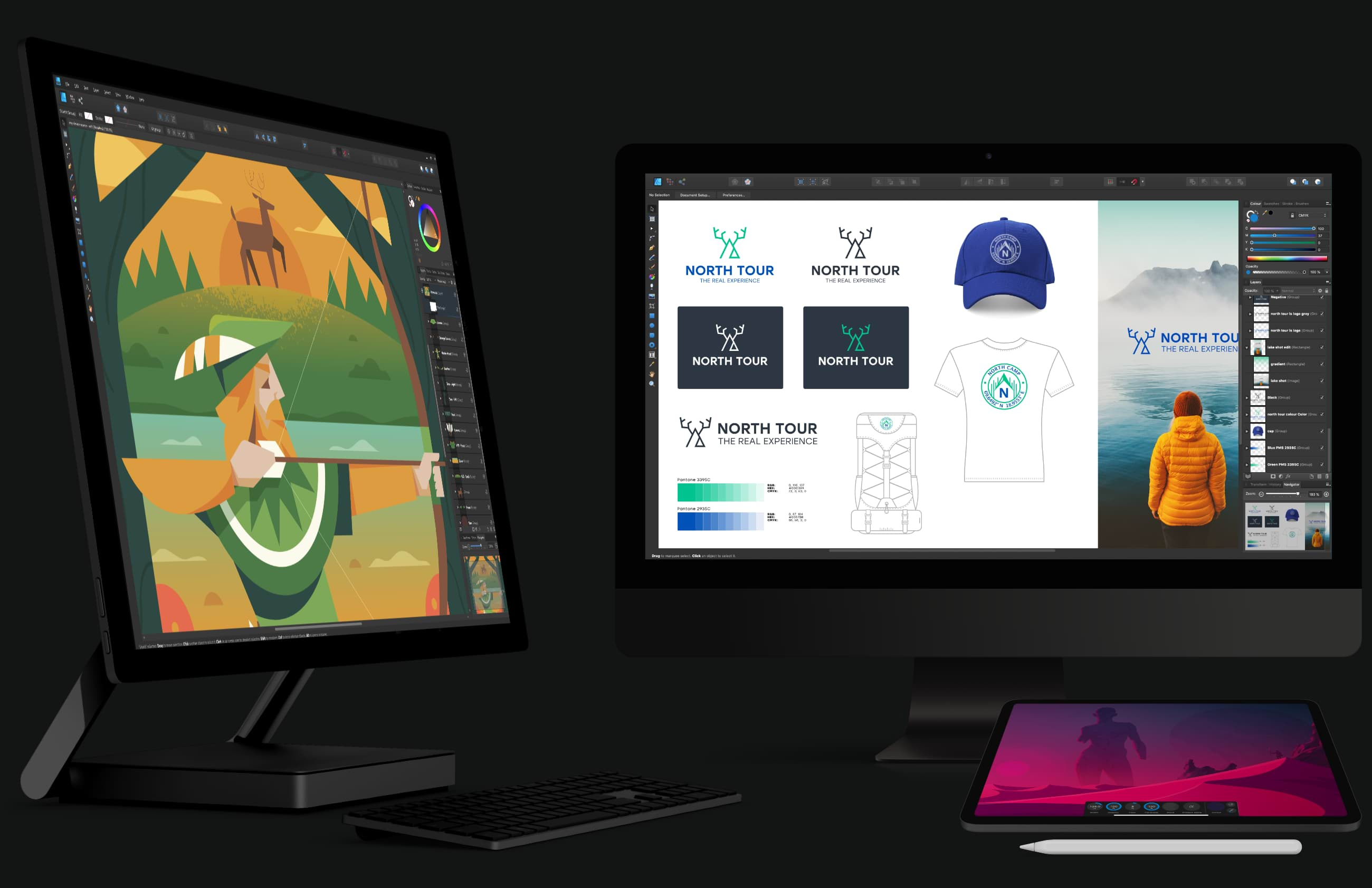
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | JAVA NEATBEANS NetBeans IDE es un entorno de desarrollo integrado de código abierto y gratuito para el desarrollo de aplicaciones en los sistemas operativos Windows, Mac, Linux y Solaris. El IDE simplifica el desarrollo de aplicaciones web, empresariales, de escritorio y móviles que utilizan las plataformas Java y HTML5. Además, ofrece soporte para el desarrollo de aplicaciones PHP y C/C++.  NetBeans IDE ofrece herramientas de primer nivel para el desarrollo de aplicaciones móviles, de escritorio, empresariales y web Java. Es el primer IDE que admite las últimas versiones de JDK, Java EE y JavaFX. Brinda resúmenes inteligentes que le ayudan a comprender y administrar sus aplicaciones, incluida la compatibilidad inmediata con tecnologías populares, como Maven. Gracias a sus características de desarrollo de aplicaciones integrales, la mejora constante de Java Editor y las mejoras continuas de velocidad y rendimiento, NetBeans IDE marca el ritmo para el desarrollo de aplicaciones con tecnologías novedosas listas para usar. |  | GUIA MATERIAL- VISTA, MODELO Y CONTROLADOR |  |
|  |  |  |
|  |
|  |  |  |  |
|  |  | Adso 3 2504452  SANTIAGO ANDRÉS LOZANO M.  dainer fabian bolano carrecedo.  SENA  2022 |  |
|  |  |  |  |
|  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | CONTROLADOR **La clase** es la esencia de Java. Es la base sobre la cual se construye todo el lenguaje Java porque la clase define la naturaleza de un objeto. Como tal, la clase forma la base para la programación orientada a objetos en Java. Dentro de una clase se definen los datos y el código que actúa sobre esos datos.  **Las clases** en Java (Java Class) son plantillas para la creación de objetos. ​ Como tal, la clase forma la base para la programación orientada a objetos en Java, la cual es una de los principales paradigmas de desarrollo de software en la actualidad. ​ Dentro de una clase se definen los datos y el código que actúa sobre esos datos. El código está contenido en métodos. Debido a que las clases, los objetos y los métodos son fundamentales para Java. Tener una comprensión básica de estas características le permitirá escribir programas más sofisticados y comprender mejor ciertos elementos claves de Java. |  | VISTA **JFrame** es una clase utilizada en Swing (biblioteca gráfica) para generar ventanas sobre las cuales añadir distintos objetos con los que podrá interactuar o no el usuario. A diferencia de JPanel, JFrame posee algunas nociones típicas de una ventana como minimizar, cerrar, maximizar y poder moverla. |  |  |
|  |  |
|  |  |  | MODELO **Los getters y los setters** se utilizan para acceder a los atributos de forma controlada, asi ninguna clase puede modificar desde fuera de los mismos los valores de dichos atributos, es un principio de encapsulación, al mismo tiempo, puedes codificar ciertas condiciones o realizar cambios en los datos de entrada antes de pasarlos al atributo de la clase. **El constructor** se utiliza para definir que sucede en el momento que creamos una instancia de una clase mediante el operador new,el constructor nos permite asignar valores por defecto a los campos o llamar a otro constructor o metodo si asi se desea. |
|  |  |  |  |  |
|  |  |  |
|  |  |

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

***VISTA***

Descripción Tratada.

El proyecto Denominado Proceso Sistematizado para el Control de Venta y almacenamiento, destaca en el Aparte de la **VISTA** 2 JFrame Form nombrados: **LOGIN** y **Sistema,** en el cual en el transcurso de esta documentación de Word estaríamos observando con mayor claridad y síntesis.

***VISTA: LOGIN***

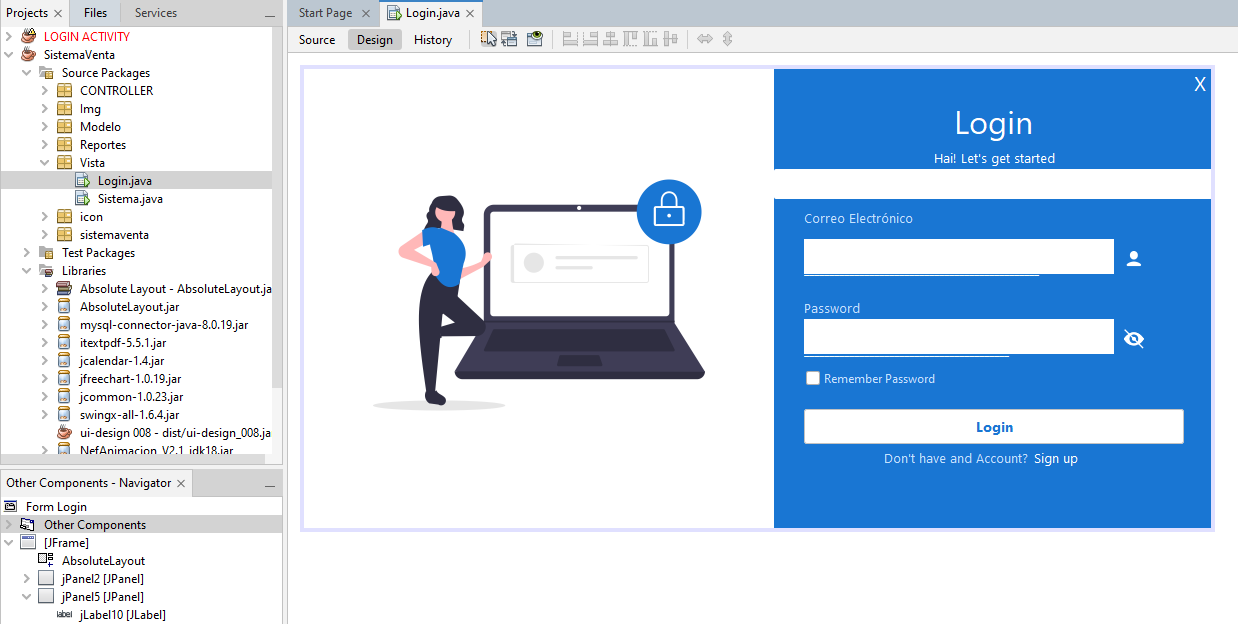
DEscrIPCIÓN DEL JFRAME FORM: LOGIN

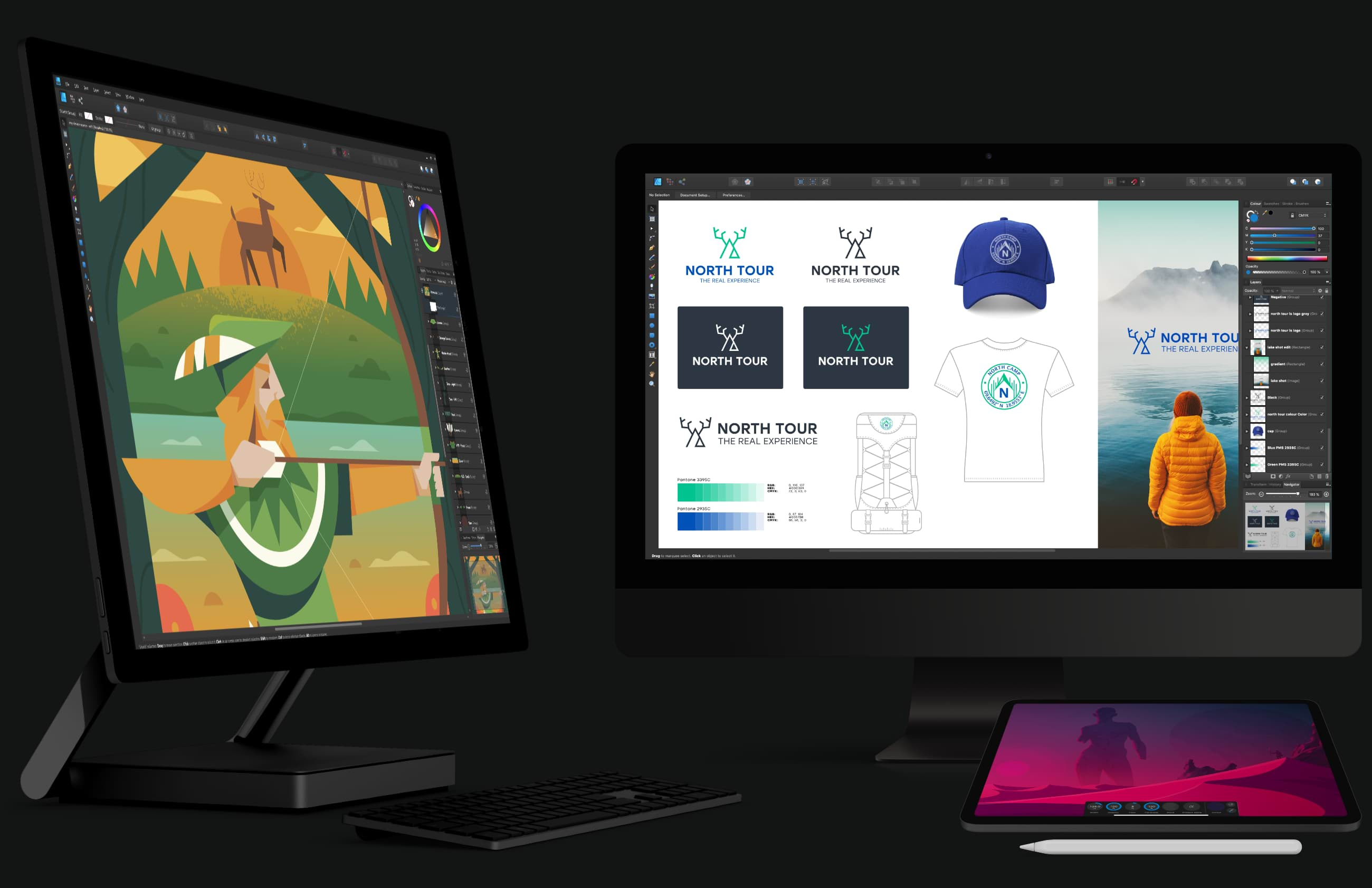
Lleno de funcionalidades útiles, eficiente, robusto, estéticamente atractivo, intuitivo, bien integrado con otros programas del entorno, para los amantes del un proceso Visionado y Moderno.

Campos abiertos para el ingreso de datos que serán almacenado gracias a la Distribución de un paquete libre denominado XAMPP, en el cual manejados Y controlados con el lenguaje de programación llamado: MYSQL.

Si se te olvido lo que digitas o no sabes lo que ingresaste, ¡No te preocupes!, EXISTE una opción para visualizar los campos digitados dentro de la contraseña, en el cual este marcado con un círculo de color ROJO. ¡CIERTO QUE SEGURIDAD Y DETALLES A LA VEZ!

Una vez ingresado el Correo Electrónico y la Contraseña, en la cual debe estar registrada en la base de datos. Se abrirá un proceso de carga [Representado por el color amarillo en la imagen mostrada], y a continuación accederá al Sistema [JFRAME FORM], en cual es el Main o el Corazón del Proyecto. Podríamos definirlo entre otros aspectos, pero que Mejor que hacerlo en otro documento, la Sintaxis del Sistema de forma INDIVIDUAL. ¡Eso es!





DESCRIPCIÓN DEL JFRAME FORM: sISTEMA

A veces El Diseño no es solo como se ve y se siente, el diseño es cómo funciona.

Dentro del Apartado Sistema, existe un Menu Desing UI, tipo Sliding capaz de pasar Opciones por opciones sin que la Información se pierda, gracias nuevamente a la Distribución de un paquete libre denominado XAMPP, en el cual manejados Y controlados con el lenguaje de programación llamado: MYSQL.

Definiendo las Opciones que tiene este JFRAME FORM: SISTEMA, contiene 7, denominados: Nuevas ventas, clientes, proveedores, productos, ventas, configuración y usuarios.

Definiendo la Opción: Nuevas Ventas.

En la imagen mostrada Contempla cada uno de los requisitos que se necesitan a la hora de hacer un venta que contiene diversas información necesaria que serán digitados por el usuario y automáticamente rellanados por el sistema, en la cual será **registrada en la fecha hoy**[actual], acompañado de una gráfica de reporte de ventas, ¡Un Sistema que cualquieeeer Vendedor Necesita! , **sumándole la capacidad de Imprimir la venta Registrada**.

Pero la majestuosidad del Proceso sistematizado aún no termina, existen otras 5 opciones que son dependiente de esta Opción: Nuevas ventas, en el cual estaremos describiéndolo a detalles su funcionalidad.

***VISTA: Sistema***

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

DESCRIPCIÓN DEL JFRAME FORM: sISTEMA- Customers

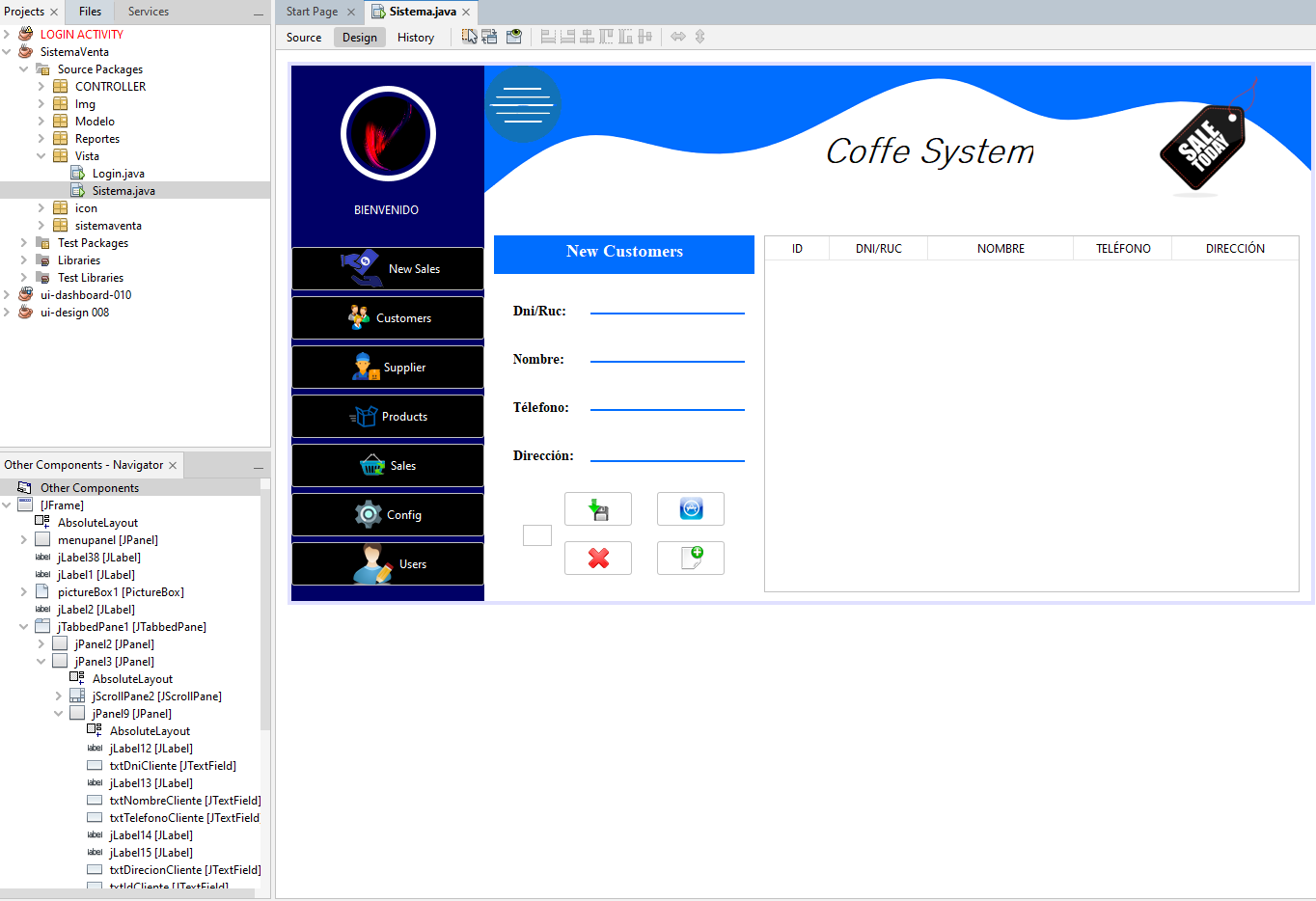
Algunos de nuestros fuertes del proyecto es brindarle la prioridad ingresos de nuevos clientes en nuestras ventas, así como también la información almacenada gracias a la Distribución de un paquete libre denominado **XAMPP**, en el cual manejados Y controlados con el lenguaje de programación llamado: **MYSQL……MYSQL , XAMPP esta bien.**

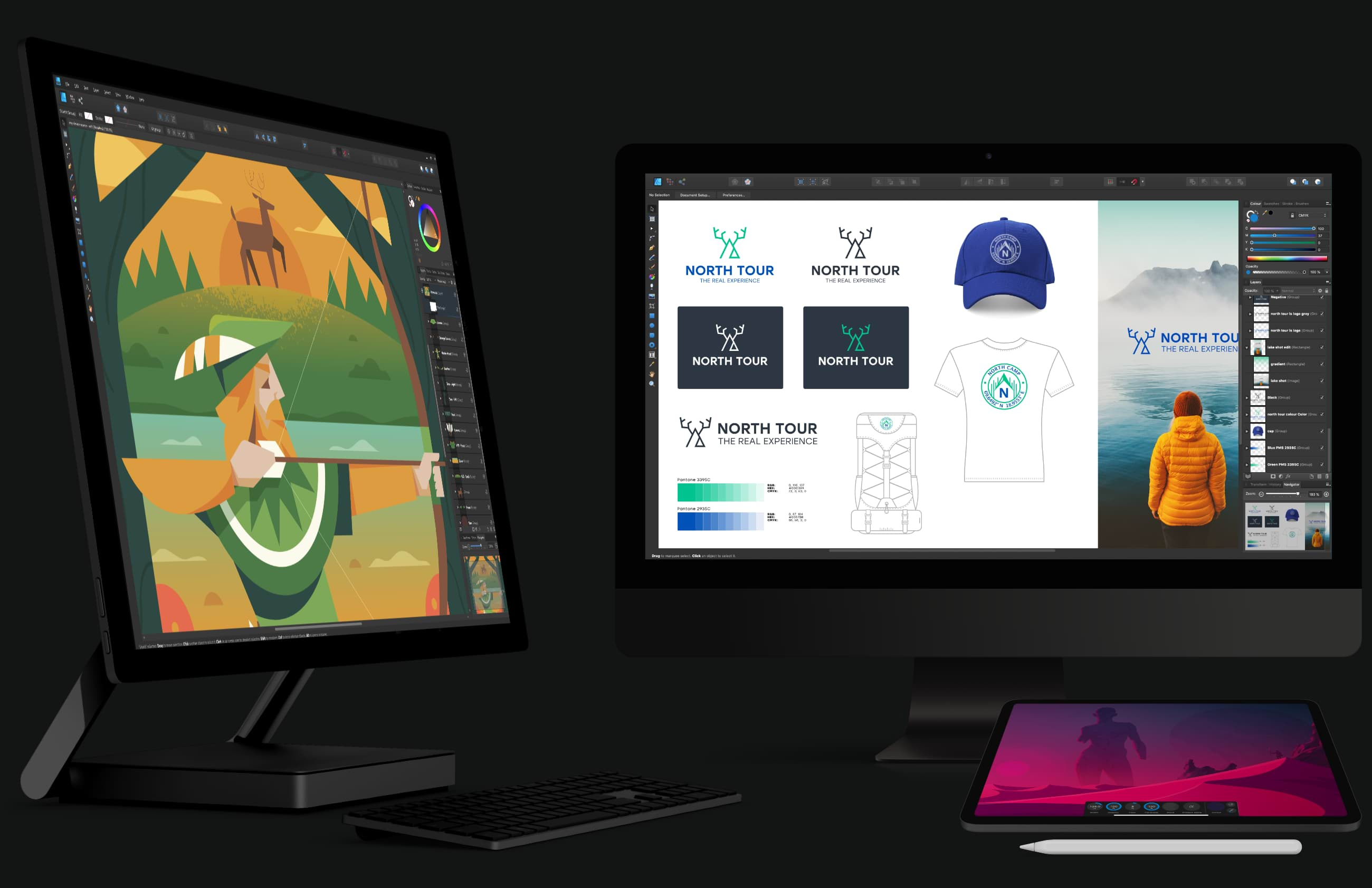
En la opción New Customers, el usuario interesado en hacer una compra que será seleccionado por el mismo vendedor, rellenará los datos, así como también, modificarlos, eliminarlos y buscarlos. Una vez digitado e ingresado, eliminado o modificado, no solo se guardar en el **MYSQ**L sino también **en la** **tabla** que se encuentra al lado del registro.

Si hay clientes…Hay proveedores. ¡CLARO QUE SI!

Tenemos **una Opción llamada: Supplier** que se encuentra dentro de menú, **con otras 4 opciones** restantes que aun estaríamos hablando a detalle en este documento. Sin más agregar en este apartado, ¡SIGAMOS!

***VISTA: Sistema- Customers***





DESCRIPCIÓN DEL JFRAME FORM: sISTEMA- supplier

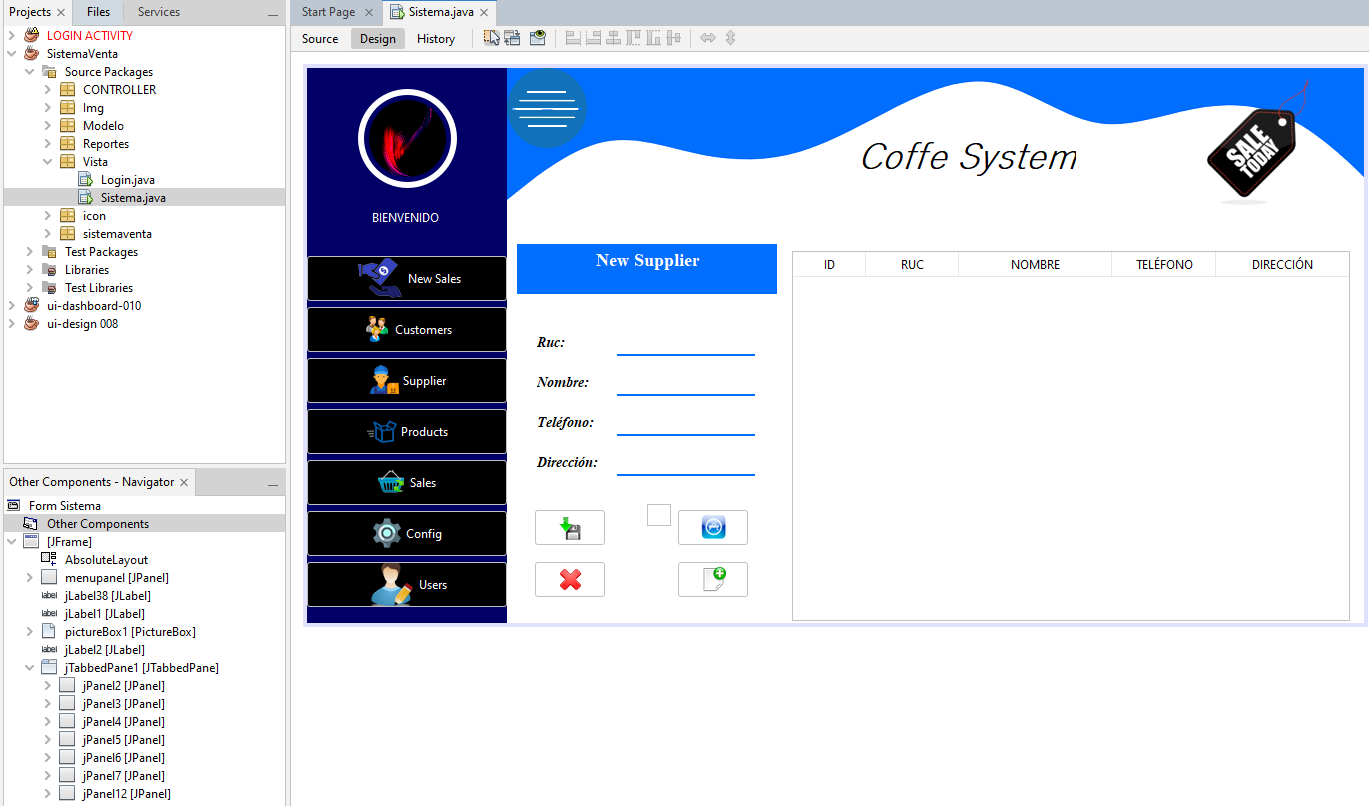
Prioridad aquellos que nos brindan prioridad como a Nosotros como vendedores.

Es algo Cansable ver las mismas opciones que contiene este Menú, Sin embargo siempre suele ser importante para nosotros enseñarle todo, hasta lo repetible.

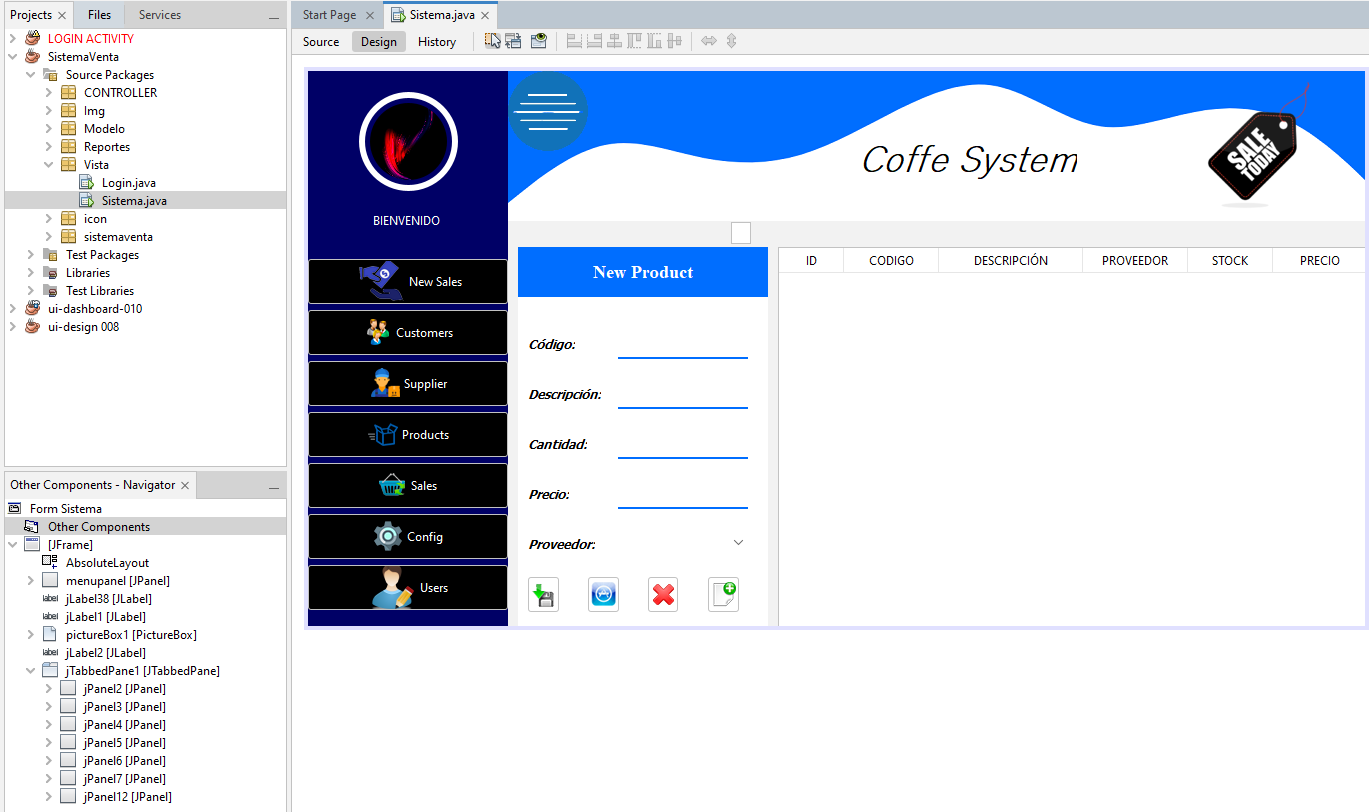
En esta opción: **Supplier** contiene los mismos detalles que posee la Opción: **Clientes**, **unas Funcionalidades de Guardar, eliminar, modificar y buscar que será almacenada**, recordando por la 3ra vez, gracias a la Distribución de un paquete libre denominado **XAMPP**, en el cual manejados Y controlados con el lenguaje de programación llamado: **MYSQL, Además agregándolo a la tabla presente en la imagen enseña.**

**Sin embargo, tenemos ¿Quién sería un Proveedor Sin productos? Tenemos otra Opción Llamada: Products que se encuentra en encuentro dentro del menú, con otras 3 opciones por mencionar. Sigan Observando.**

***VISTA: Sistema-Supplier***



***VISTA: Sistema-Products***

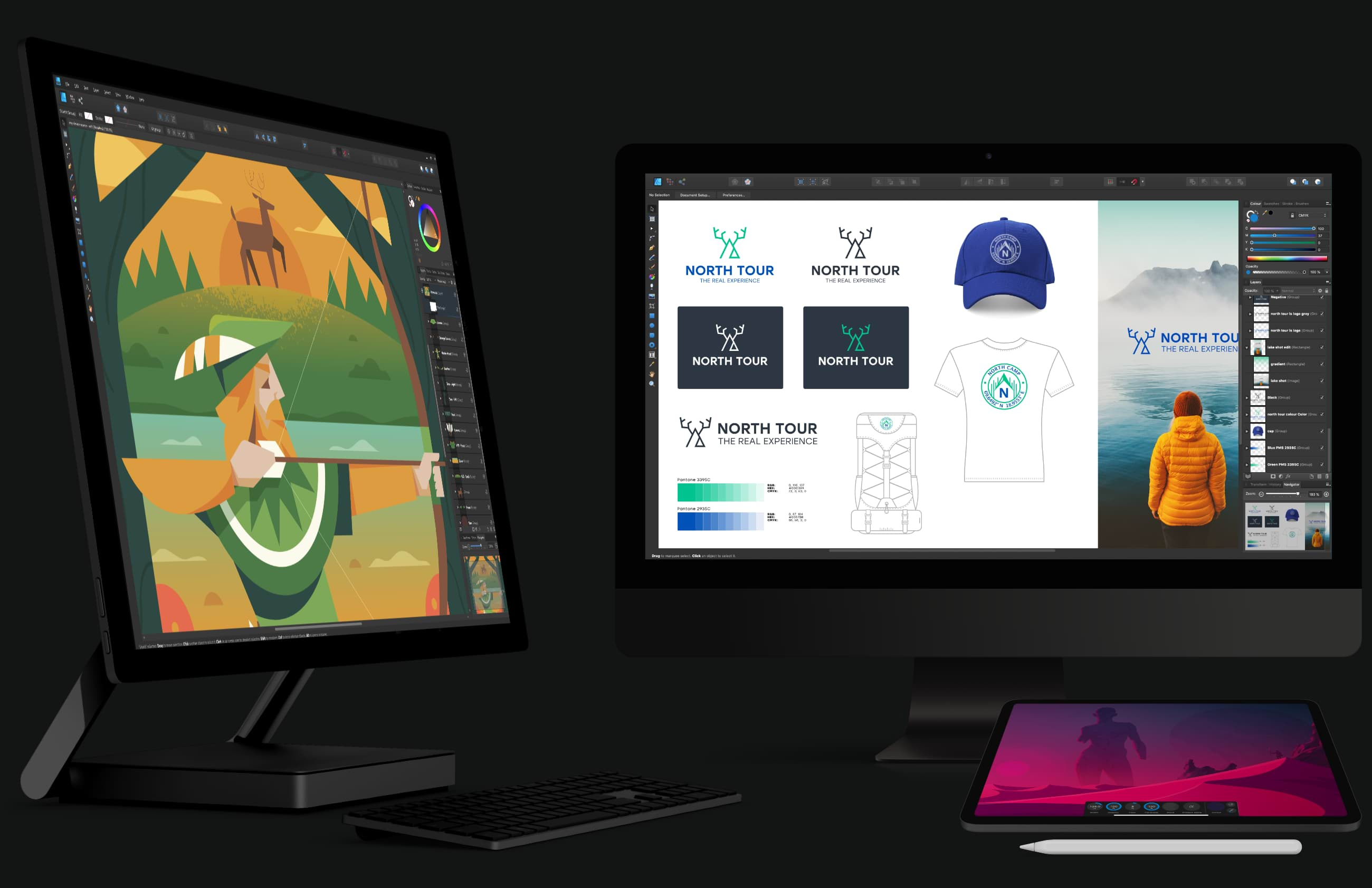


DESCRIPCIÓN DEL JFRAME FORM: **SISTEMA- PRODUCTS**

¡Productos al Almacén! De eso es lo que se trata como acto fundamental Nuestro Proyecto.

Las funcionalidades Son las mismas juntos con el ya es costumbre mencionar la Distribución de un paquete libre denominado **XAMPP**, en el cual manejados Y controlados con el lenguaje de programación llamado: **MYSQL, Además agregándolo a la tabla presente en la imagen enseña.**

**Sin Embargo, existe algo que destaca esta Opción, y es el ComboBox[Marcado de color azul],** en el cual se cargaran los proveedores, siempre y cuando este registrados en el Sistema y en el Base de datos. Muy Curioso y Necesario, Así como también el Registro completo de todas las ventas que se realicen en el Sistema, obviamente depende de la Opción New Sales. La dependencia que se encuentra dentro del menú, con otras 3 opciones por agregar en este documento. Nos acercamos al principio de la Funcionalidad, pero dentro, El Backend.



DESCRIPCIÓN DEL JFRAME FORM: **SISTEMA- Sales**

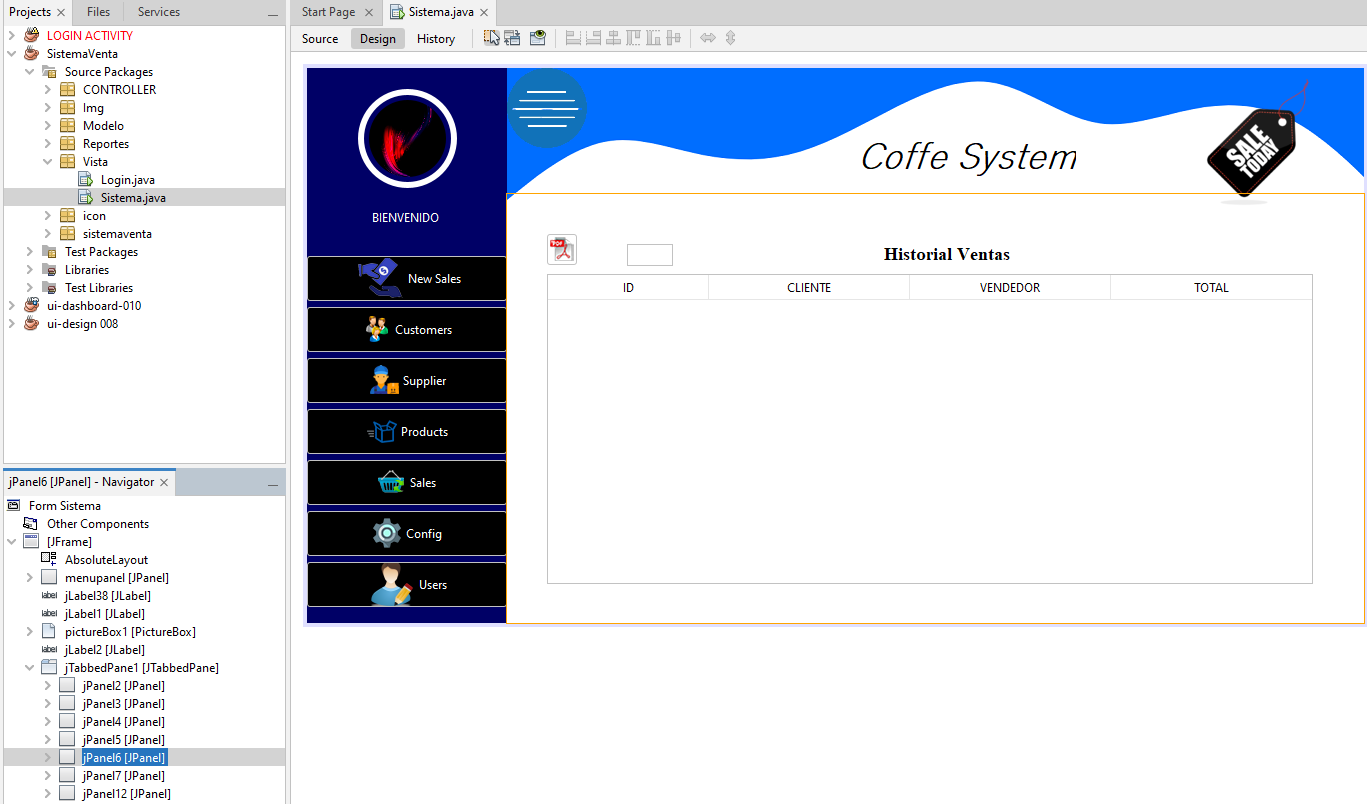
En el Sistema todo es posible desde un grafico de las ventas registrada hasta una historia de todas ventas, Sin contar un aspecto importante…

En ese Panel el historial de todas las ventas es gracias a la Distribución de un paquete libre denominado **XAMPP**, en el cual manejados Y controlados con el lenguaje de programación llamado: **MYSQL.**

Una vez agregado algún elemento así sea uno o mas dentro la tabla, el manejador del Sistema tendrá la oportunidad **agregar ese historial mencionado a un Archivo Tipo PDF**, en el cual le otorgara más fluidez, control y mucho manejo en la venta.

Hay un aspecto que se me había olvidado mencionar en este documento, definimos que los proveedores tienen dependencia con los productos, **pero si hablamos de que también existe una dependencia entre proveedores y compañía**. ¡ES POSIBLE EN NUESTRO PROYECTO!

***VISTA: Sistema-Sales***

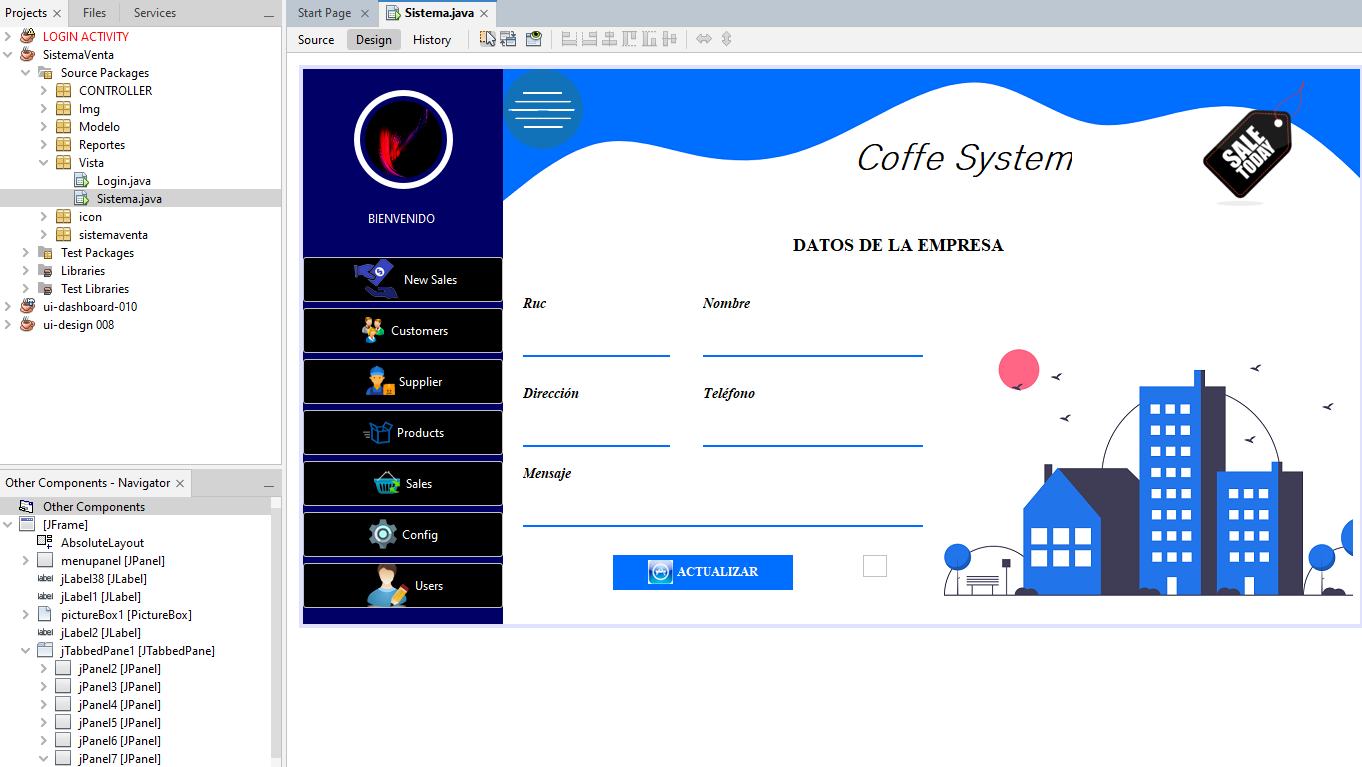


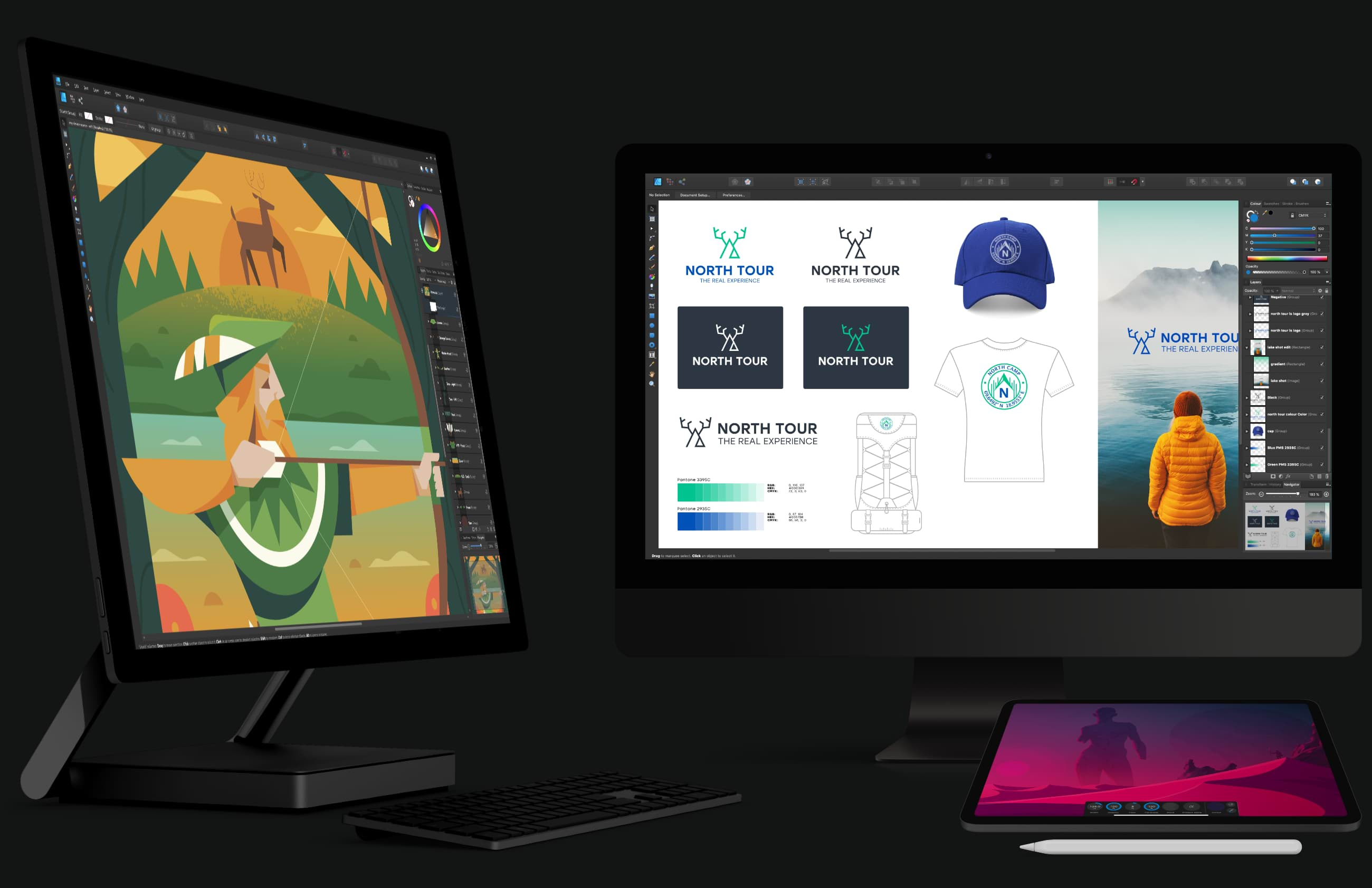
DESCRIPCIÓN DEL JFRAME FORM: **SISTEMA- Config**

Y Aun seguimos con los registros, pero dependencia, entre el proveedor y la compañía.Sin embargo, si hablamos de dependencia, esta no esta conectada a la base, sino que obtiene los datos que se registran en el proveedor.

La eficacia es otro punto destacable, y aprovechando el uso del término, una vez actualizado los datos dentro de esta opción: Config se guardar y se mantendrán registrada hasta nuevamente sea modificada. Sin embargo, si no existe una información registrada, está por defecto mantendrá todos sus datos como valor “Null”

***VISTA: Sistema-Config***





DESCRIPCIÓN DEL JFRAME FORM: **SISTEMA- USERS**

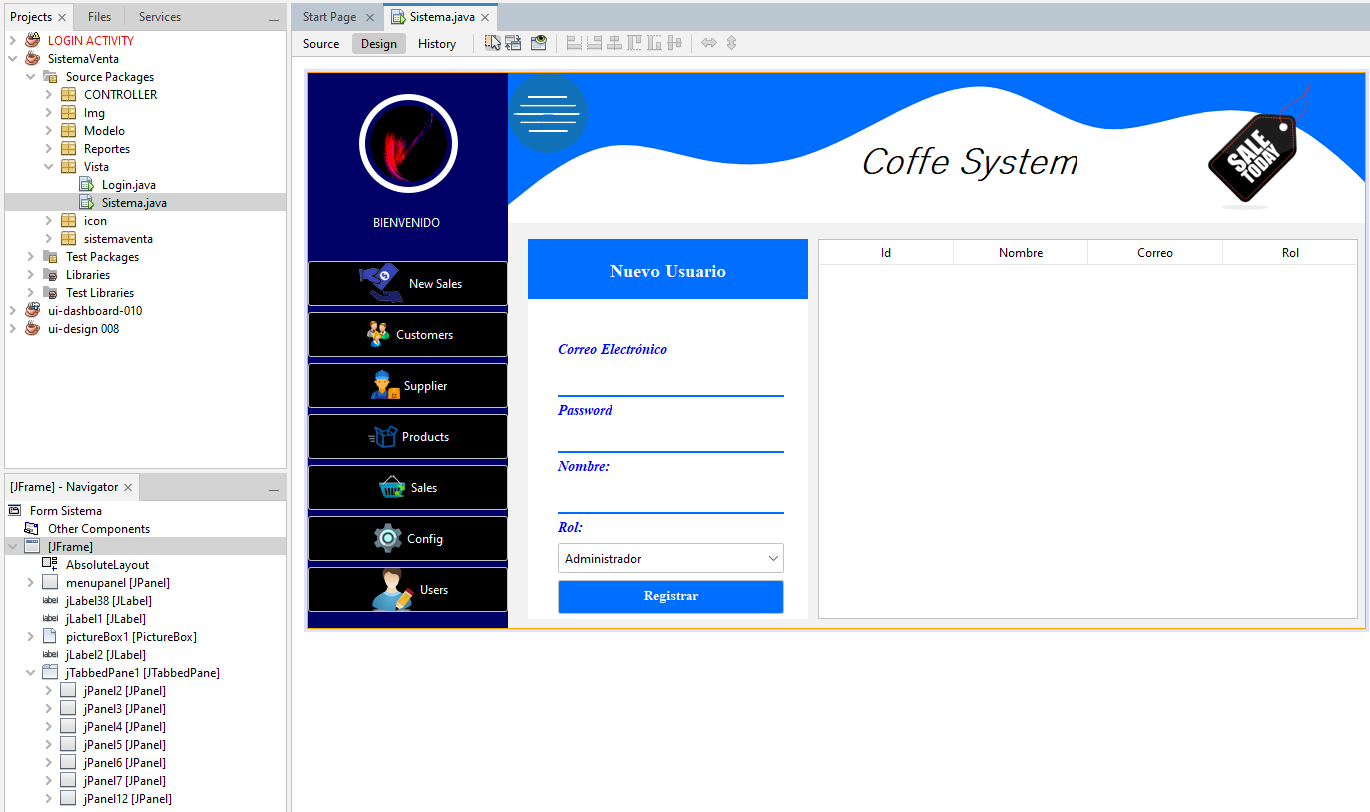
Y como ultima vista dentro del Sistema, y mas importante en el aspecto de seguridad de ingreso al sistema es la Opción: **Users.**

Esta vista permite únicamente Agregar nuevos Usuarios con su respectivo registro que contiene: **Correo Electrónico, Contraseña, Nombre y muy el rol**, encargado de identificar que tipo de usuario se registra dentro del sistema, porque si no existiera esta opción **¿Qué Usuarios están accediendo al sistema de ventas por medio del registro del Login y la Base de Datos dentro del Coffe System?**

Todo este registro, ya sabrán que es graciasla Distribución de un paquete libre denominado **XAMPP**, en el cual manejados Y controlados con el lenguaje de programación llamado: **MYSQ, además un tabla que proyecta la información digitada. En el cual se encarga de validar en el LOGIN, si el usuario si existe dentro de la base de datos o no.**

**Y hasta aquí Acaba la Vista. Una nuevo proceso Backend Comienza , con un mundo amplio atrás de ese diseño**

***VISTA: Sistema-Users***



***Modelo-Conexión***

* Descripción

Llevando a java-Neatbeans a otro nivel, Conector a base de Datos.

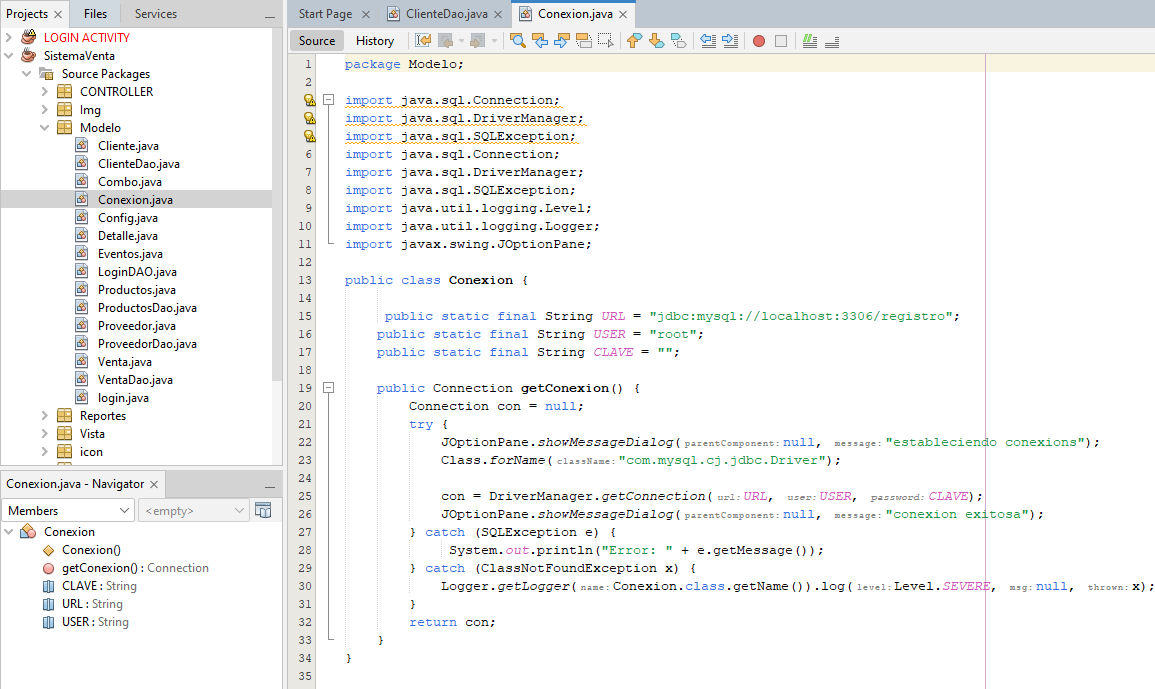
Importamos algunos componentes que serán necesarios para la conexión, incluyendo en la Librería un .JAR Connector MYSQL 80.12.

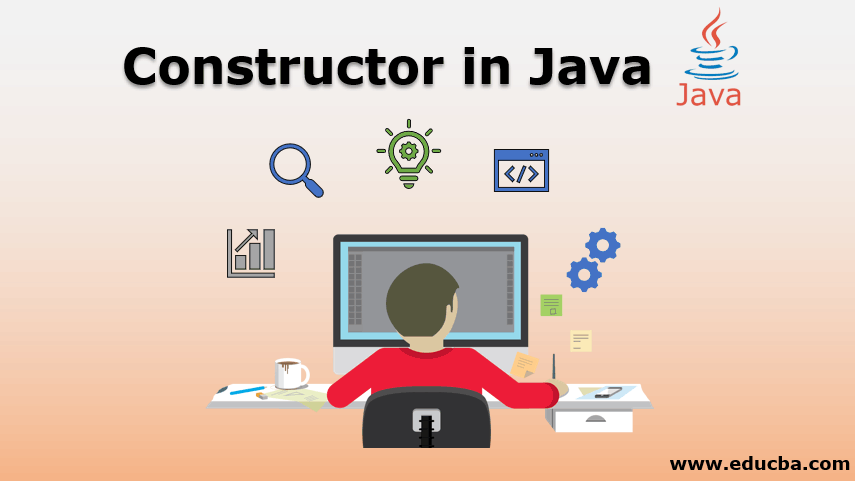
Creamos Una clase publica llamada **“CONEXIÓN”,** estableceremos valores públicos estáticos, URl es la dirección de nuestra base, con el puerto pertinente y el nombre de la Base de datos que vamos a almacenar los datos.

Nuevamente creamos otra clase publica llamada **“Connection”** “**GetConexion”,** en el cual es una variable que necesita ser importada.

Estableceremos Un Valor **“Connection”** denominado **“Con”** como valor **“Null”**

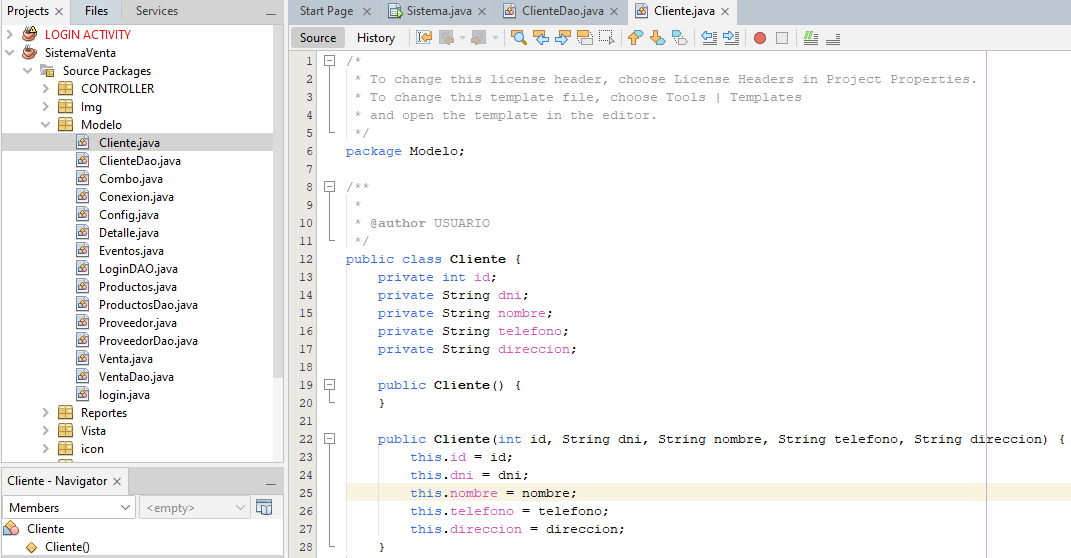
**Usaremos a continuación un Try-Catch para detectar y controlar un excepción generada por el funcionamiento, en el cual importaremos otro valor para el drive, de siguiente un driver conexión para establecer la URL, USER, y Clave, Si es correcto nos arroja el JOptionPane, si no un error y nos indica el problema**



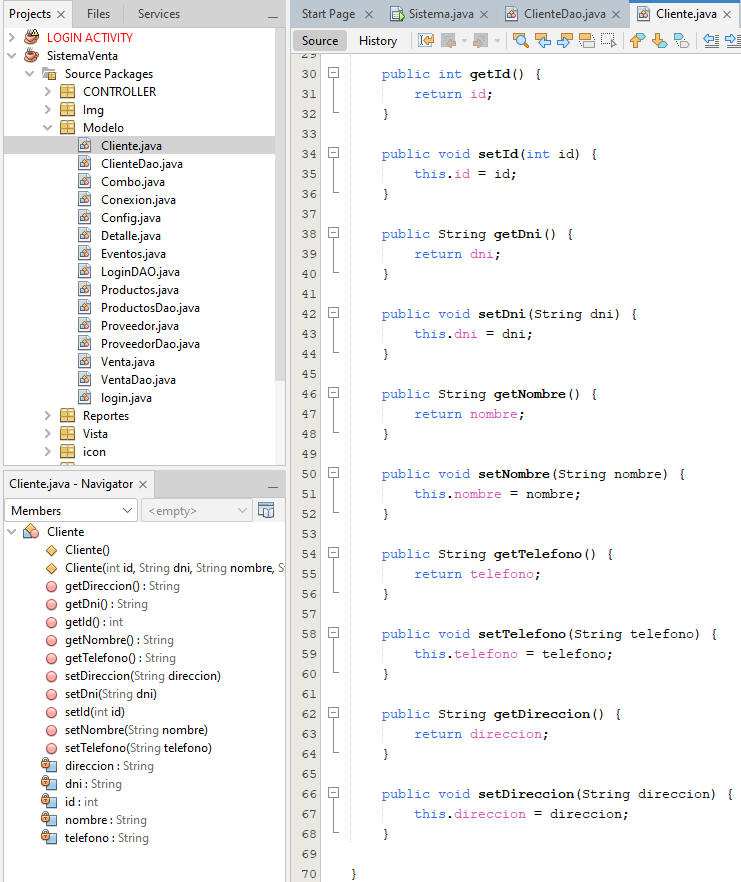


***Modelo Cliente***

CONSTRUCTOR



Getter And Setter



Descripción

Descripción Modelo-**ClienteDato:**

Creamos una clase denominada “**ClienteDao**” en el cual tendrá como “**instancia** “**Conexión**”, “**Connection**”, “**PreparedStatement**” y “**ResultSet**”

Creamos Otro valor public Boolean, este se encargará saber si el dato es verdadero o falso y retornarlo. Agregando como parámetro el Modelo Cliente con nombre **“Registrar Cliente**”

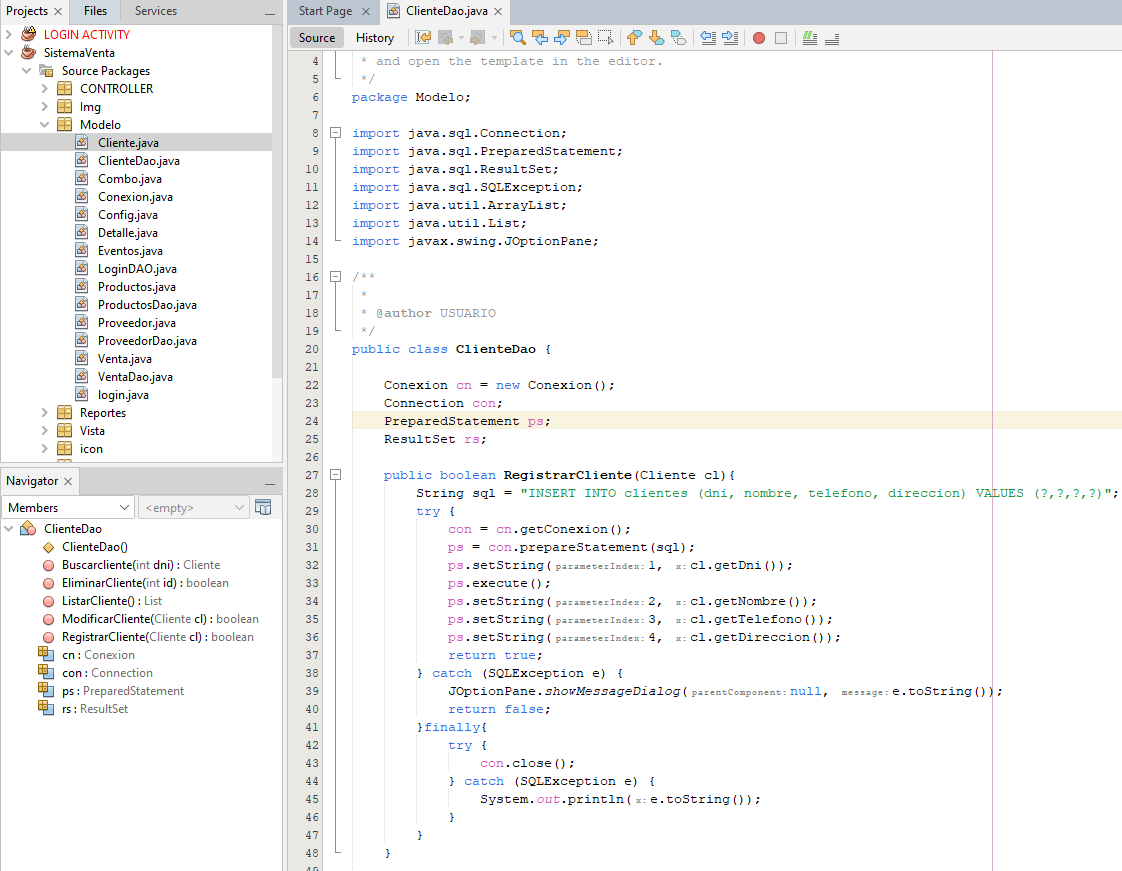
Establecemos la Consulta tipo String sql en el cual insertaremos los atributos mencionados con valores que van agregarse.

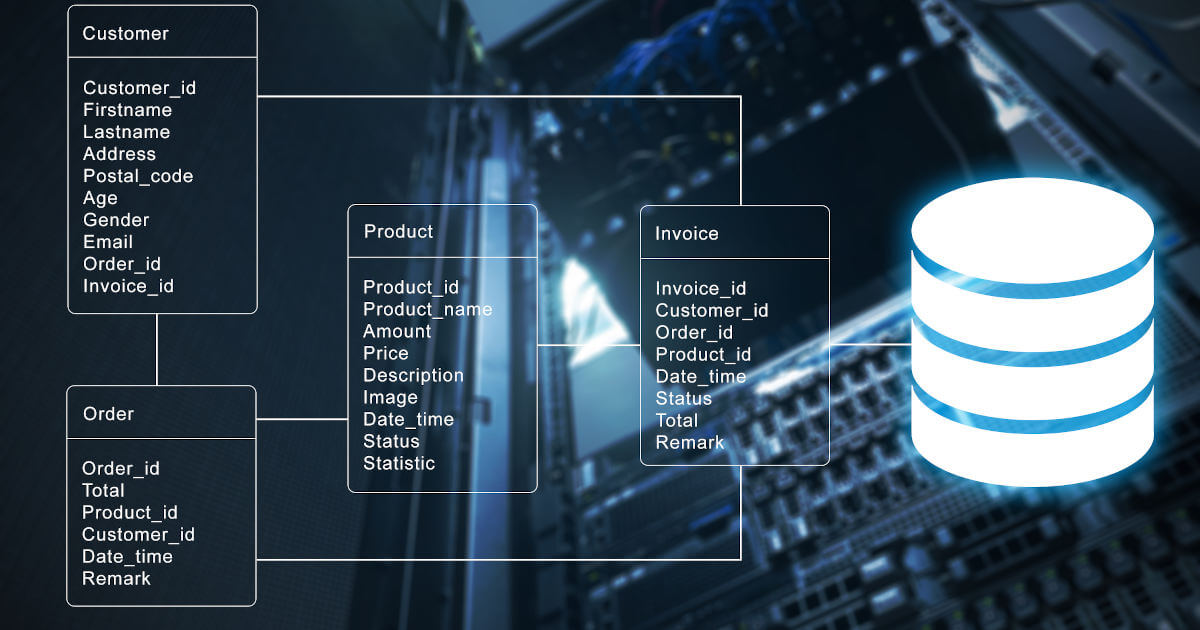
Usamos un Try para detectar, y de seguido aplicamos los valores que instanciamos en la clase publica **“Cliente Dao”, con sus respectivas obtenciones de datos que agregamos en el modelo y que servirán para almacenarlo y agregarlo de tipo string en la base de datos en forma de posición. Si no funciona usamos el Joption para capturar el error , lo retornar nuevamente, y por ultimo un valor “Finally” Para cerrar la operación y no se mantenga abierta, ejecutándose.**

***Modelo ClienteDao***

***Registrar Clientes***

***Modelo Cliente Dao***





Descripción

Creamos una Publica lista llamada “**ListarCliente”** usaremos un array list para almacenarlo en una lista, necesario para una tabla. Lo importamos.

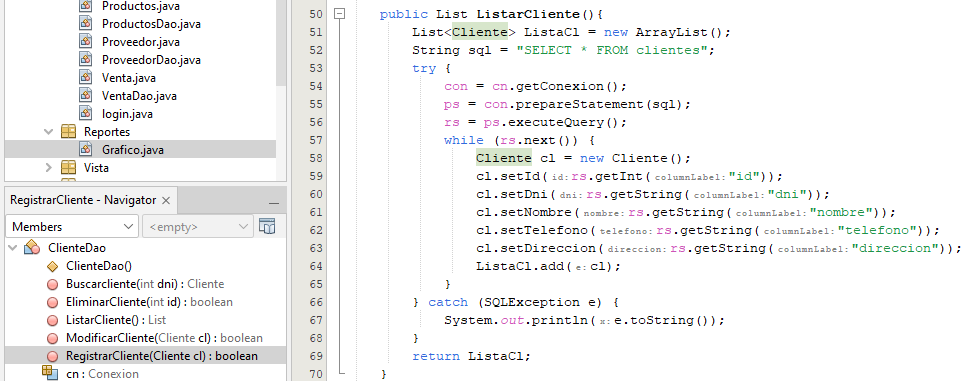
De seguido establecemos la consulta tipo String sql en el cual Seleccionaremos todos los atributos de la Tabla **“Clientes”.**

Usamos un Try para detectar, y de seguido aplicamos los valores instanciamos en la clase publica **“Cliente Da” ,** usamos un “While” para repetir las sentencias, y denominamos el valor Cliente de la Lista como “C1”.

Ese valor que nosotros nombramos, se aplicado en los Modelo de los Clientes que se enviaran a los atributos correspondiente que le asignemos, y luego añadidos a la tabla. Un “Catch” Para evaluar y luego nuevamente retornar la lista del array denominado “C1”

***Modelo ClienteDao***

***Listar Clientes***



A

***Modelo ClienteDao***

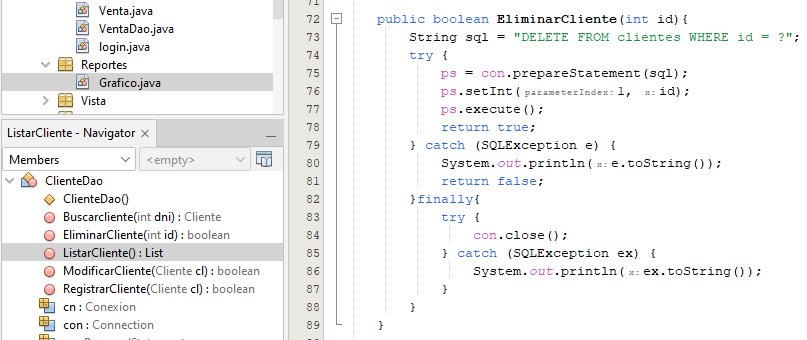
***Eliminar Clientes***

Descripción

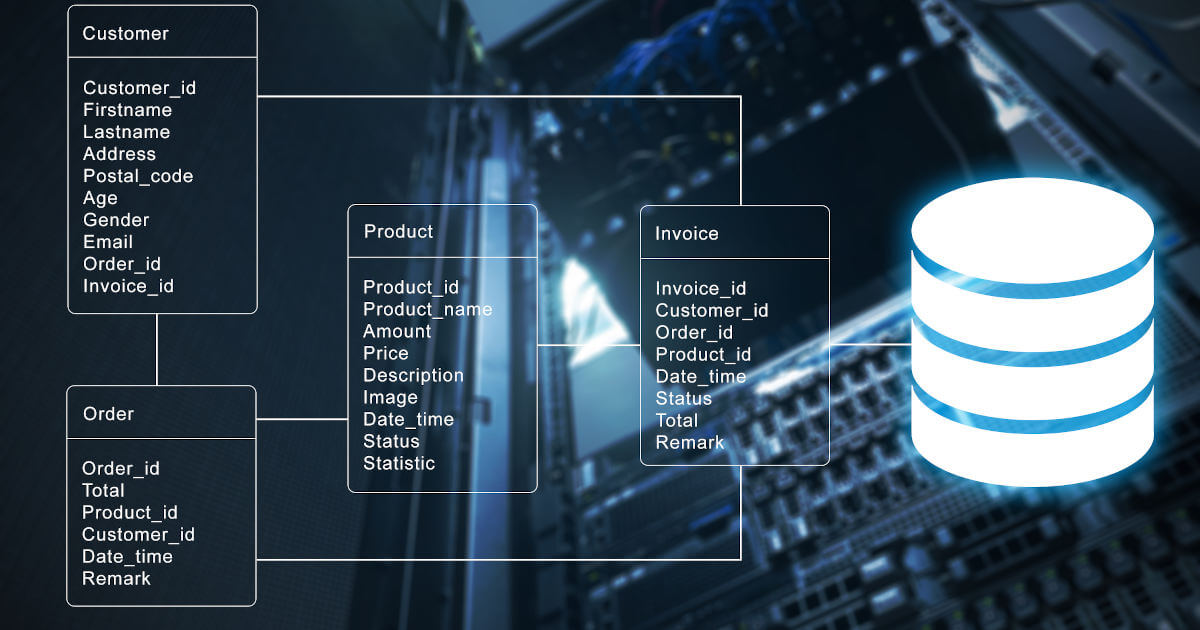
Creamos un public boolean[Comprueba verdadero y falso] de parámetro enterio de ID, ya que ese va el parámetro que se a utilizar para borrar cliente, y como id es llave primaria, el valor es irrepetible, genera un valor único.

De seguido establecemos la consulta tipo String en el cual Eliminaremos de la tabla Cliente un valor id no especifico.

Usamos un “Try” para detectar, y de seguido aplicamos los valores instanciamos en la clase publica, agregándole un nuevo valor de tipo “SetINT”, es decir que se va a aplicar ese valor de forma id entero en posición 1, y como la llave primaria va autoincrementado ese valor, buscamos el dato hasta llegar con él y eliminar mediante su id, sino ocurre esto un “Catch” Para detectar el problema. A lo último, un **“Finally”** Para cerrar la búsqueda y no se mantengan ejecutándose.



A



***Modelo ClienteDao***

***Modificar Clientes***

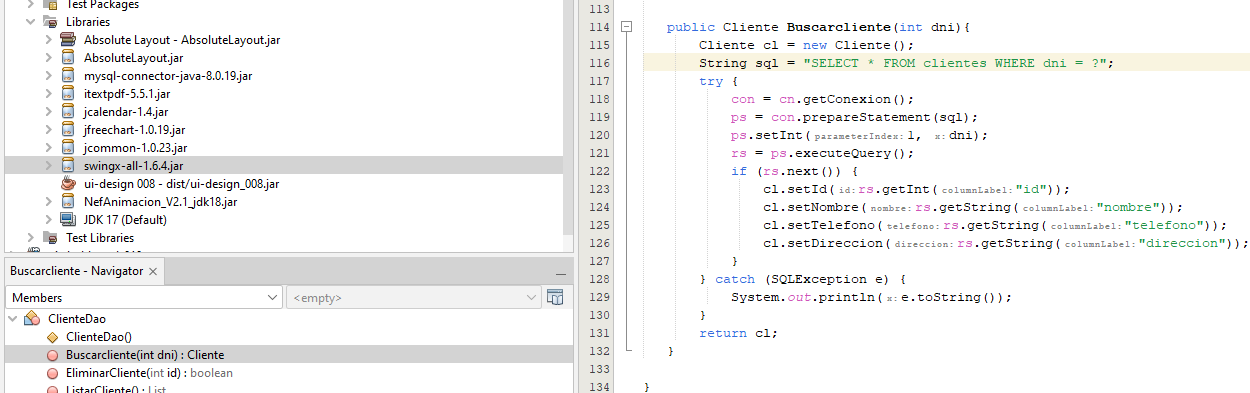


Descripción

Creamos Nuevamente un **“public boolean”[**Comprueba verdadero y falso], luego establecemos la consulta tipo String e el cual modificaremos algunos de sus atributos por medio de id del cliente, ya que el id es un valor irrepetible de auto\_increment.

Usamos un **“Try”** para detectar, y de seguido aplicamos los valores que instanciamos en la clase publica, con el almacenamiento de los constructores y os getter and setter, podemos almacenar esos datos. Retornar esa valor, por consiguiente un **“Catch”** para evaluar los errores y al final un **“FINALLY”**  para cerrar la consulta y no se mantenga en ejecución.

A



A

A

A

A

A

A

A

A

A

A

A

A

A

A

A

A

A

A