Índice

Introducción

Las aplicaciones informáticas, son el pivote de muchas transacciones comerciales en la vida económica, muchas empresas utilizan herramientas computacionales para poder realizar reportes, cálculos, controles, etc. de una gran diversidad de productos, un mejor control de sus trabajadores información relevante a los negocios y muchas de ellas se han convertido en parte fundamentales para la toma de decisiones. Como por ejemplo si un empresario quiere saber qué producto que vende o produce le provee mayor ganancia, cuantos ha producido, que utilidad le han generado en ciertos periodos de tiempo por lo cual se hacer estas herramientas informáticas son fundamentales e indispensables en la actualidad.

En este caso se nos encargó el desarrollo de una aplicación sencilla de utilizar y con una interfaz amigable para con el usuario, el fin primordial del proyecto es la elaboración de un sistema donde se pueda manejar información en archivos de texto, la meta principal proyectada de la aplicación es la elaboración de reportes en base a la información de los archivos de texto. Se planifica que la solución al proyecto expuesto sea con el uso exclusivo de estructuras de datos genéricas y estructuras de datos creadas específicamente para cada tipo de modulo conjunto de datos.

Objetivos:

- Crear una interfaz sencilla pero no por eso simple que permita guardar los datos concernientes a las ventas, empleado, productos, clientes.
- Crear y acceder a una base de datos que contenga la información relevante que se almacena y poderla consultar posteriormente.
- Usar una estructura de tipo de datos dinámica específica para guardar la información.

Análisis:

Como se indicó anteriormente las aplicaciones informáticas son el pivote de muchas transacciones en la actualidad. Se desarrollara una aplicación que permita agilizar los procesos según la actividad de la empresa en este caso ventas, empleado, producto y clientes.
Para el proceso ventas: se necesita que en este módulo se pueda hacerse registros de las ventas realizadas.
Para el proceso empleado: se necesita que en este módulo se pueda registrar datos con las características de una persona, realizar búsqueda de datos previamente registrados y la eliminación del mismo.
Para el proceso producto: se necesita que en este módulo se pueda registrar datos con las características de un producto comercial código de barras, nombre una breve descripción del mismo y el precio.
Para el proceso cliente: se necesita que en este módulo se pueda registrar datos con las características de una persona cliente, nombre, número de identificación tributaria, y código designado en la empresa.

Análisis de Requerimientos de Usuario:

Se realizó una consulta con diferentes usuarios para que nos indicara de qué manera se le facilita acceder a las diferentes opciones de una aplicación informática. Según estas consultas los usuarios indicaron lo siguiente:

- No les gusta una interfaz con fondo negro pantalla negra.
- Que los comando no se tengan que escribir tipo consola.
- Que al momento de surgir un problema el error mostrado no contenga información muy técnica ya que no la comprenden.
- Independientemente de lo fácil que sea de utilizar la herramienta informática exista un manual de uso.

Planteo inicial de la solución:

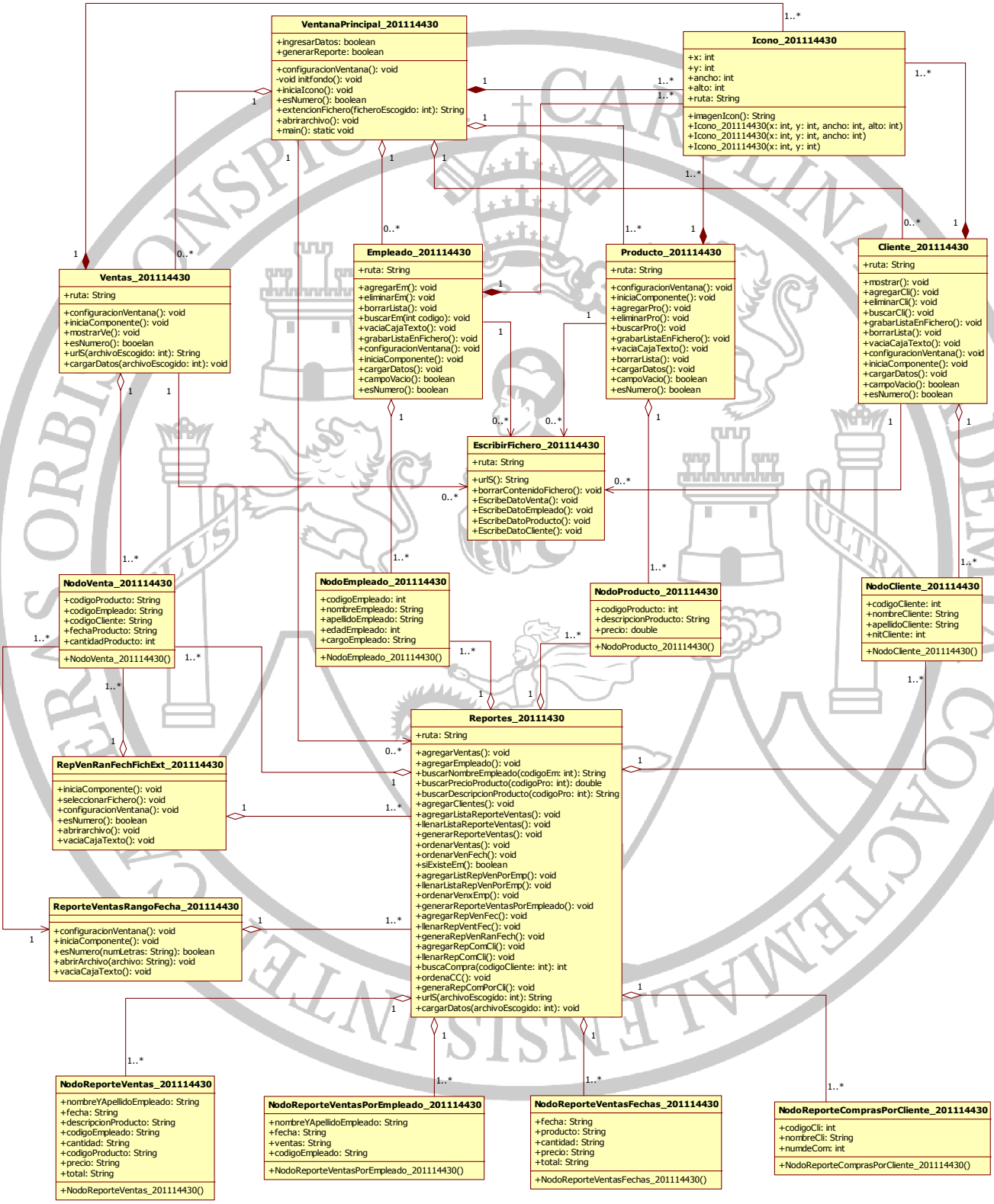
Para no limitar la ejecución de la aplicación en un solo entorno (Sistema Operativo) distribución con núcleo Linux, Mac OSX o Microsoft Windows se utilizara para el desarrollo de la aplicación un lenguaje multiplataforma robusto y estable en este caso se utilizara java.

Según las especificaciones dadas se hace evidente diseñar y desarrollar cuatro tipos de clases las cuales encapsularan los datos específicos concernientes a los procesos que controlan o registran.

Para la generación de los reportes en este caso cuatro diferentes tipos de reporte se hace evidente por las especificaciones dada de la persistencia de datos que se debe realizar como mínimo consulta a dos tipos de base de datos y como máximo a tres bases de datos diferentes. Al realizar este proceso de generación de reportes se hace necesaria la creación de otras cuatro clases las cuales encapsularan los datos específicos consultados de las diferentes bases de datos.

Para la administración de estas clases anteriormente mencionadas que encapsulan los datos se crearan cuatro clases una por cada clase según su proceso.

Diagrama de Clases:



Descripción Detallada por clase:

Clase VentanaPrincipal_201114430:

Esta clase contiene los componentes principales de interacción con el usuario.
Contiene el método main el cual inicia el flujo de la aplicación.
Contiene los siguientes métodos:
void configuracionVentana():contiene la configuración del marco de la ventana principal.
void initfondo(): hace la instancia a la clase Icono_201114430 y le envía al constructor el parametron "fondo" para establecer un fondo a la aplicación.
void iniciaIcono(): establece los parámetros de construcción de las etiquetas e iconos principales y los iconos para generar reportes de fuentes externas.
boolean esNumero(String numLetras): este método recibe un parámetro String numérico. En el cual se utiliza el método matches para revisar que la cadena String sea un número permitido.
String extencionFichero(int archivoEscogido): este método recibe un número del uno al cuatro y segun el numero regresa la extensión del archivo que se debe seleccionar. Lo manda a llamar el filtro creado para seleccionar únicamente los archivos permitidos.
void abrirarchivo(String archivo): este método abre el documento html creado en cada reporte.

Clase Icono_201114430:

Esta clase hereda de JButton contiene los parametros de los iconos que se utilizan, contiene tres constructores una para los iconos grandes, uno para los iconos medianos y uno para establecer el fondo de la ventana principal. Contiene los siguientes:
Final String ruta: variable pública que almacena la ubicación predeterminada de los archivos base de datos.
String imagenIcon(): Metodo que recibe como parámetro un número y retorna la dirección de una imagen.

Clase Ventas_201114430:

Esta clase construye una lista de las ventas registradas utiliza el la clase NodoVentas_201114430, contiene los siguientes métodos:
void mostrarVe() Método que recorre la lista que contiene las ventas registradas y las muestra.
boolean esNumero(String numLetras) Método que recibe un numero convertido en String y llama al metodo matches para verificar que los datos ingresados por los usuarios contenga números.
String urlS(int archivoEscogido) Método que recibe un número y retorna la dirección de un fichero.
void cargarDatos(int archivoEscogido) En este método se crea instancias de File, FileReader, BufferedReader para cargar fichero desde el disco duro y almacenarlo en memoria dinámica en este caso el fichero VENTA.fct.

Clase Empleado_201114430:

Esta clase construye una lista de los empleados registrado permitiendo almacenar grabar y eliminar registro.
Contiene los siguientes métodos:
void agregarEm(): Metodo que agrega un Nuevo registro de tipo empleado a la lista.
void eliminarEm(int codigoEmp) Metodo que permite eliminar un registro previamente buscado.
void borrarLista() Metodo que borra el contenido de la lista para evitar duplicación de registros.
void vaciaCajaTexto() Metodo que establece en null las cajas capturadoras de datos JTextField.
void buscarEm(int codigoEmp) Metodo que recibe el codigo del empleado y realiza una busqueda.
void grabarListaEnFichero() Metodo que guarda la lista en el disco duro (persistencia de datos).
boolean campoVacio(String contenidoCaja) Metodo que verifica antes de guardar o buscar que no haga falta ningún campo requerido.
boolean esNumero(String numLetras) Metodo que verifica que los JTextField contengan numero.
void cargarDatos(int archivoEscogido) busca el archivo necesario para cargarlo a memoria dinámica en este caso el fichero EMPLEADO.emp.

Clase Producto 201114430:

Esta clase construye una lista de los productos registrados permitiendo almacenar, grabar y eliminar registro.
void agregarPro(): Metodo que agrega un Nuevo registro de tipo empleado a la lista.
void eliminarPro (int codigoPro) Metodo que permite eliminar un registro previamente buscado.
void borrarLista() Metodo que borra el contenido de la lista para evitar duplicación de registros.
void vaciaCajaTexto() Metodo que establece en null las cajas capturadoras de datos JTextField.
void buscarPro (int codigoPro) Metodo que recibe el código del producto y realiza una búsqueda.
void grabarListaEnFichero() Metodo que guarda la lista en el disco duro (persistencia de datos).
boolean campoVacio(String contenidoCaja) Metodo que verifica antes de guardar o buscar que no haga falta ningún campo requerido.
boolean esNumero(String numLetras) Metodo que verifica que los JTextField contengan número.
void cargarDatos(int archivoEscogido) busca el archivo necesario para cargarlo a memoria dinámica en este caso el fichero PRODUCTO.prt.

Clase Cliente_201114430:

Esta clase construye una lista de los clientes registrados permitiendo almacenar, grabar y eliminar registro.
void agregarCli(): Metodo que agrega un Nuevo registro de tipo cliente a la lista.
void eliminarCli (int codigoCli) Metodo que permite eliminar un registro previamente buscado.
void borrarLista() Metodo que borra el contenido de la lista para evitar duplicación de registros.
void vaciaCajaTexto() Metodo que establece en null las cajas capturadoras de datos JTextField.
void buscarCli(int codigoCli) Metodo que recibe el código del cliente y realiza una búsqueda.
void grabarListaEnFichero() Metodo que guarda la lista en el disco duro (persistencia de datos).
boolean campoVacio(String contenidoCaja) Metodo que verifica antes de guardar o buscar que no haga falta ningún campo requerido.
boolean esNumero(String numLetras) Metodo que verifica que los JTextField contengan número.
void cargarDatos(int archivoEscogido) busca el fichero necesario para cargarlo a memoria dinámica en este caso el fichero CLIENTE.clt .

Clase NodoVenta_201114430: Esta clase es el contenedor de los datos específicos de los que se generan al realizar y registrar una venta. Su constructor recibe los siguientes parámetros:
codigoProducto de tipo entero.
codigoEmpleado de tipo entero.
codigoCliente de tipo entero.
fechaProducto de tipo String.
cantidadProducto de tipo entero.

Clase NodoEmpleado_201114430: Esta clase es el contenedor de los datos específicos de los empleados. Su constructor recibe los siguientes parámetros:
codigoEmpleado de tipo entero.
codigoEmpleado de tipo entero.
nombreEmpleado de tipo String.
apellidoEmpleado de tipo String.
edadEmpleado de tipo entero.
cargoEmpleado de tipo String.

Clase NodoProducto_201114430: Esta clase es el contenedor de los datos específicos de los producto. Su constructor recibe los siguientes parámetros:
codigoProducto de tipo entero.
descripcionProducto de tipo String.
precio de tipo doublé.

Clase NodoCliente_201114430: Esta clase es el contenedor de los datos específicos de los clientes. Su constructor recibe los parámetros siguientes:
codigoCliente de tipo entero.
nombreCliente de tipo String.
apellidoCliente de tipo String.
nitCliente de tipo entero.

Clase EscribirFichero_201114430: Esta clase contiene cuatro métodos para guardar las listas de memoria dinámica al disco duro, siendo los siguientes:
void escribeDatoVenta() recibe los parámetro de la lista ventas.
void escribeDatoEmpleado() recibe los parámetro de la lista empleados.
void escribeDatoProducto()recibe los parámetro de la lista productos.
void escribeDatoCliente()recibe los parámetro de la lista clientes.
String urlS(int archivoEscogido) recibe un número y retorna el nombre y dirección del archivo.

Clase Reporte_201114430: Esta clase contiene las listas ventas, empleado, producto y clientes, interactúa con todas y genera los reportes. Las listas anteriormente mencionadas contienen los mismos métodos de las clases Ventas_201114430, Empleado_201114430, Producto_201114430, Cliente_201114430.

Clase NodoReporteVentas_201114430 Encapsula los datos para el reporte de ventas para guardarla en el archivo Ventas.html.
Clase NodoReporteVentasPorEmpleado_201114430 Encapsula los datos para el reporte de ventas por empleado para guardarla en el archivo VentasxEmpleado.html.
Clase NodoReporteVentasFechas_201114430 Encapsula los datos para el reporte de ventas por fechas para guardarla en el archivo VentasRangoFechas.html.
Clase NodoReporteComprasPorCliente_201114430 Encapsula los datos para el reporte de Compras por cliente para guardarla en el archivo ComprasporCliente.html.

Clase ReporteVentasRangoFecha_201114430 en esta clase se ingresa fecha inicial y fecha final hace una instancia de la lista ventas y muestra las que están dentro del rango establecido.

Persistencia de Datos:

Archivo venta:

Expresión Regular para separar ","

Codigo Empleado.	Empleado	Fecha.	Codigo Producto.	Descripción Producto.	Cantidad.	Precio.	Total.

1, 2.3.2015, 1, 2, 3

Archivo Empleado:

Expresión Regular para separar ","

Codigo.	Nombre	Apellido.	Edad.	Cargo.

1, Martin, Solares, 24, Vendedor

Archivo Producto:

Expresión Regular para separar ","

Codigo.	Descripcion	Precio.

1, Zapatos, 300

Archivo Cliente:

Expresión Regular para separar ","

Codigo.	Nombre	Apellido.	Nit

1, Pedro, Gonzalez, 4176495

Glosario de términos:

Persistencia de Datos:

De modo técnico, se refiere a la propiedad de los datos para que estos sobrevivan de alguna manera.

Nodo:

Concretamente en estructuras de datos, un nodo es uno de los elementos de una lista enlazada, de un árbol o de un grafo. Cada nodo será una estructura o registro que dispondrá de varios campos, y al menos uno de esos campos será un puntero o referencia a otro nodo, de forma que, conocido un nodo, a partir de esa referencia, será posible en teoría tener acceso a otros nodos de la estructura.

Memoria Dinámica:

La memoria dinámica se refiere a aquella memoria que no puede ser definida ya que no se conoce o no se tiene idea del número de la variable a considerarse.

Conclusión:

Al desarrollar la aplicación (el proyecto final) se hizo notoria la necesidad de separar las diferentes fases de la aplicación lectura, escritura. Para generar los reportes era necesario guardar los datos a nivel de memoria dinámica y luego en la memoria no volátil (disco duro) de forma permanente.