**TRABAJO PRÁCTICO INTEGRADOR**

**Materia**: Algoritmos y estructuras de datos I

**Integrantes**: Cao Ivan, Hanisch Andres, Sica Franco y Olazar Maximo

**Nombre del Equipo**: El Koto



**Propuesta:** El proyecto consiste en la creación de un Software para un Supermercado, que abarque distintas áreas, enfocado para el uso en Caja como para uso Administrativo en cuanto a diversas funciones.

El mismo contará con un menú interactivo para que el usuario pueda:

* Consultar información de productos (precio, promociones, stock, descripción, ubicación física)
* Alta, baja y modificación de productos.
* Calculadora integrada para realizar las diferentes operaciones en caja. A su vez, estará conectada con la base de datos de los productos, por lo que disminuirá su stock, así como devolverá el total a pagar, y el cambio a entregar.
* Creación de estadísticas generales, como cantidad de ventas, facturación, productos más vendidos.

**Funcionalidades Implementadas:**

1. **Menú Principal**:
   * El programa presenta un menú interactivo que permite al usuario navegar por las opciones.
   * Opciones disponibles:
     + Consultar información de productos.
     + Alta, baja y modificación (ABM) de productos.
     + Funciones de caja.
     + Generación de estadísticas.
     + Salir del programa.
2. **Consulta de Información de Productos**:
   * Permite visualizar los productos almacenados en un archivo JSON.
   * Muestra datos como precio, promociones, stock y ubicación física del producto.
3. **ABM de Productos (Alta, Baja, Modificación)**:
   * **Agregar Productos**: Se pueden añadir nuevos productos al sistema con todos los detalles requeridos.
   * **Modificar Productos**: Es posible editar los detalles de un producto ya existente.
   * **Eliminar Productos**: Los productos se pueden eliminar del sistema de manera permanente.
4. **Manejo de Archivos JSON**:
   * Los datos de los productos se almacenan en un archivo JSON.
   * Para esto, se importo la librería json de Python, la cual permite leer, modificar y guardar la información de los productos en el archivo json.
   * Se incluyen funciones para gestionar los archivos JSON, que almacenan todos los productos con sus detalles.
5. **Caja**:
   * Conexión con la base de datos de productos para realizar operaciones de venta.
   * Reducción automática de stock tras cada operación y cálculo del cambio.
6. **Salida del Programa**:
   * Una opción que muestra un banner de despedida y finaliza la ejecución.

**Funcionalidades Planeadas:**

1. **Generación de Estadísticas**:
   * Herramientas para crear reportes sobre las ventas, facturación y productos más vendidos.

**Guía de usuario:** Explicación de las funciones usadas en el Software.

**menu\_principal():**

* **Banner**: Se define una cadena de texto llamada banner que contiene un arte ASCII que se muestra al inicio. Este banner añade un toque visual atractivo al menú.

**Impresión del Menú**: La función imprime las opciones disponibles para el usuario:

**1. Consultar información de productos**

**2. ABM Productos** (Agregar, Buscar, Modificar productos)

**3. Caja**

**4. Estadísticas**

**5. Salir**

**Entrada del Usuario**: La función solicita al usuario que ingrese una opción del menú.

* Se asegura de que la opción ingresada sea un número entero y esté dentro del rango de 1 a 5.
* Ingrese nuevamente una opción correcta.

**Valor de Retorno**: Finalmente, la función devuelve la opción seleccionada por el usuario, que se puede usar posteriormente en el programa para ejecutar la acción correspondiente.

**leer\_archivo():**

**Intenta abrir el archivo**: La función intenta abrir un archivo llamado productos.json en modo lectura ("r"). Este archivo se supone que contiene datos de productos en formato JSON.

**Lectura del archivo**: Si el archivo se abre correctamente, se lee su contenido completo y se almacena en la variable lineas\_productos.

Luego, el archivo se cierra para liberar el recurso.

**Verificación de contenido**: La función verifica si el contenido del archivo está vacío (longitud de lineas\_productos es 0). Si es así, imprime un mensaje indicando que el archivo está vacío o no contiene datos, y termina la ejecución de la función sin devolver nada.

**Carga de datos JSON**: Si el archivo tiene contenido, se utiliza json.loads para convertir la cadena JSON en un objeto de Python (generalmente una lista o diccionario). Este objeto se almacena en la variable productos.

**Valor de Retorno**: Finalmente, la función devuelve el objeto productos, que puede ser utilizado más adelante en el programa para manipular los datos leídos del archivo.

**Manejo de Errores**: Si ocurre algún error durante la apertura o lectura del archivo, la función captura la excepción y muestra un mensaje de error indicando que no se puede abrir el archivo productos.

**leer\_archivo\_ventas():** Está diseñada para leer los datos de un archivo JSON que contiene información sobre ventas y devolver estos datos en una estructura de Python (lista de diccionarios, por ejemplo).

**Intento de abrir el archivo**: Se utiliza un bloque try-except para intentar abrir el archivo ventas.json en modo lectura ("r").

**Lectura del contenido**: Si se abre correctamente, la función lee todo el contenido del archivo mediante archivo.read().

**Verificación de contenido vacío**: Si el archivo no contiene ningún dato (longitud del contenido es 0), muestra un mensaje de advertencia y finaliza la ejecución de la función.

**Cierre del archivo**: Después de leer el contenido, el archivo se cierra con archivo.close().

**Conversión a JSON**: Si el archivo tiene contenido, este se convierte de una cadena JSON a una estructura de datos Python mediante json.loads(), y se devuelve el objeto.

**Manejo de errores**: Si ocurre algún error durante el proceso (como si el archivo no existe), se muestra un mensaje de error y la función termina sin devolver nada.

**menu\_info\_productos():**

**Llamada a leer\_archivo**: La función comienza llamando a leer\_archivo() para cargar los datos de productos desde el archivo productos.json. Los datos se almacenan en la variable productos.

**Obtención de las columnas**: Se obtiene una lista de las claves (nombres de las columnas) del primer elemento en la lista de productos usando productos[0].keys().

Esto permite saber qué datos se van a mostrar en la tabla.

**Cálculo de anchos de columnas**: Se crea un diccionario anchuras que almacena el ancho máximo de cada columna. El ancho se calcula comparando la longitud del nombre de cada columna con la longitud de los valores de cada fila en esa columna.

Se recorre cada fila de productos, actualizando el ancho de cada columna según sea necesario.

**Impresión de la cabecera**: Se imprime la cabecera de la tabla, alineando los nombres de las columnas según su ancho calculado.

Se genera y muestra un separador entre la cabecera y las filas de datos.

**Impresión de los datos**: Se recorren las filas de productos e imprime cada fila de datos, alineando los valores según el ancho de sus respectivas columnas.

**Menú de búsqueda**: Se presenta al usuario un menú con opciones para realizar búsquedas:

**1. Búsqueda por Nombre**

**2. Búsqueda por Marca**

**3. Búsqueda por Promoción**

**4. Volver**

Se solicita al usuario que ingrese una opción, validando que esté entre 1 y 4.

**Manejo de búsquedas**: Si el usuario elige una opción de búsqueda (1, 2 o 3), se pide el valor a buscar y se llama a la función busqueda\_filtrada, pasando la columna correspondiente y el valor ingresado.

Después de cada búsqueda, se vuelve a mostrar el menú de búsqueda.

**Finalización**: Si el usuario selecciona la opción 4 para volver, la función termina.

**busqueda\_filtrada(productos, columna\_busqueda, valor\_busqueda):**

**Obtención de las columnas**: La función comienza obteniendo las claves (nombres de las columnas) del primer elemento de la lista de productos, utilizando productos[0].keys(). Esto establece qué datos se mostrarán en la tabla.

**Filtrado de datos**: Se crea una lista llamada datos\_filtrados que contiene solo las filas de productos donde el valor\_busqueda (convertido a minúsculas) está presente en el valor de la columna especificada (columna\_busqueda).

Utiliza una comprensión de lista para recorrer cada fila y aplica str(fila.get(columna\_busqueda, '')).lower() para asegurarse de que la búsqueda no distinga entre mayúsculas y minúsculas.

**Verificación de resultados**: Si no se encuentran resultados (es decir, si datos\_filtrados está vacío), se imprime un mensaje informando que no se encontraron resultados para el valor buscado y la función termina.

**Cálculo de anchos de columnas**: Si se encuentran resultados, se calcula el ancho de cada columna, similar a cómo se hace en la función menu\_info\_productos. Esto se realiza para alinear correctamente los datos en la tabla.

**Impresión de la cabecera**: Se imprime la cabecera de la tabla alineando los nombres de las columnas según el ancho calculado. También se imprime un separador entre la cabecera y los datos.

**Impresión de datos filtrados**: Se recorren las filas filtradas (datos\_filtrados) e imprime cada fila alineando los valores según el ancho de sus respectivas columnas.

**menu\_abm\_productos():**

**Impresión del Menú:** La función comienza mostrando un menú interactivo para el usuario con las siguientes opciones:

1. Alta Producto (agregar un nuevo producto)

2. Baja Producto (eliminar un producto)

3. Modificación Producto (modificar los datos de un producto existente)

4. Volver (regresar al menú anterior)

**Entrada del Usuario:** Se solicita al usuario que ingrese una opción.

* La opción ingresada se válida para asegurarse de que esté entre 1 y 4. Si no es válida, se le pide al usuario que ingrese una opción correcta.

**Lectura de Productos:** Se llama a la función leer\_archivo() para cargar la lista de productos desde el archivo productos.json, almacenando los datos en la variable productos.

**Manejo de Opciones de ABM:** Mientras la opción elegida esté entre 1 y 3, se ejecuta la acción correspondiente:

Opción 1 (Alta Producto): Se llama a la función alta\_producto(productos) para agregar un nuevo producto.

Opción 2 (Baja Producto): Se llama a la función baja\_producto(productos) para eliminar un producto.

Opción 3 (Modificación Producto): Se llama a la función modificar\_producto(productos) para modificar un producto existente.

**Repetir el Menú**: Después de ejecutar la acción correspondiente, se vuelve a mostrar el menú de ABM Productos, y se solicita al usuario que ingrese nuevamente una opción.

* Se valida de nuevo que la opción esté entre 1 y 4.

**Finalización**: Si el usuario selecciona la opción 4 para volver, la función termina.

**busqueda\_producto():** Tiene cuatro modos de operación que permiten realizar diferentes tipos de búsquedas en la lista de productos según el propósito (alta, modificación, baja o venta).

**Modo Alta**:

* Solicita el nombre y marca del producto.
* Recorre la lista de productos y comprueba si ya existe un producto con el mismo nombre y marca.
* Si se encuentra uno igual, vuelve a solicitar los datos hasta que se ingrese un producto que no exista.
* Retorno: Devuelve el nombre y la marca del nuevo producto que se quiere añadir.

**Modo Modificación**:

* Solicita el id del producto que se quiere modificar.
* Recorre la lista de productos para encontrar el producto con el id ingresado.
* Si no se encuentra, vuelve a solicitar un id válido.
* Retorno: Devuelve el idProducto y el diccionario encontrado con los datos del producto que se va a modificar.

**Modo Baja:**

* Solicita el id del producto que se quiere eliminar.
* Recorre la lista y verifica si el id existe en algún producto.
* Si no se encuentra el id, se solicita uno nuevo hasta que el id sea válido.
* Retorno: Devuelve el idProducto y un valor booleano encontrado indicando que el producto ha sido encontrado para su eliminación.

**Modo Caja:**

* Solicita el id del producto que se quiere vender.
* Recorre la lista de productos y verifica si el id existe.
* Si no se encuentra, se solicita un nuevo id hasta que se encuentre el producto.
* Retorno: Devuelve el idProducto y el diccionario encontrado con los datos del producto vendido.

**alta\_producto(productos):**

**Introducción y Solicitud de Datos:** La función comienza imprimiendo un mensaje para solicitar los datos del nuevo producto.

* Se determina el nuevo id del producto encontrando el id máximo existente en la lista de productos utilizando la función max() y obtener\_id.

**Ingreso de Información del Producto:** Se solicita al usuario que ingrese el nombre, la marca, el precio, la ubicación y el stock del producto.

* El precio y el stock deben ser valores positivos. Se válida la entrada y se le solicita al usuario que vuelva a ingresar el dato si no es válido.

**Verificación de Duplicados:** Se verifica si el producto (basado en el nombre y la marca) ya existe en la lista de productos.

* Si el producto ya existe, se solicita al usuario que ingrese un nuevo nombre y marca hasta que se introduzca uno que no esté duplicado.

**Opciones de Promoción:** Se pregunta al usuario si desea agregar una promoción al producto. Si responde afirmativamente, se le presentan varias opciones para definir la promoción.

* Dependiendo de la opción elegida, se le solicita al usuario que ingrese los valores necesarios para crear la promoción, asegurándose de que los valores sean válidos.

**Adición del Producto:** Se crea un nuevo diccionario con todos los datos del producto (incluyendo id, nombre, marca, precio, ubicacion, stock, y promocion) y se añade a la lista productos mediante productos.append().

**Guardado en Archivo JSON:** Se convierte la lista de productos a formato JSON utilizando json.dumps() y se guarda en el archivo productos.json.

* Se maneja una posible excepción para detectar errores al abrir o escribir en el archivo, informando al usuario si no se puede grabar el archivo.

**Finalización:** Se imprime un mensaje confirmando que el producto ha sido agregado correctamente.

**obtener\_id(elemento):**

* **Propósito:** La función está diseñada para facilitar el acceso al valor del id de un elemento (en este caso, un diccionario que representa un producto).
* **Parámetro:** elemento: Se espera que sea un diccionario que contenga al menos la clave id.
* **Retorno:** La función devuelve el valor asociado con la clave id del diccionario pasado como argumento.

**baja\_producto(productos):** Permite eliminar un producto de una lista de productos almacenada en un archivo JSON, basándose en el id proporcionado por el usuario.

**Mostrar productos:** Se llama a la función mostrar\_info\_productos() para mostrar la lista de productos antes de la eliminación.

**Entrada del usuario:** Se solicita al usuario que ingrese el id del producto que desea eliminar.

**Búsqueda del producto:** Se recorre la lista de productos para verificar si existe un producto con el id ingresado.

* Si no se encuentra el producto, se le pide al usuario que ingrese otro id.

**Eliminación del producto:** Si el producto se encuentra, se crea una nueva lista (productos\_actualizados) que contiene todos los productos excepto el que se desea eliminar.

**Actualización del archivo:** La lista actualizada se convierte a formato JSON y se guarda en el archivo productos.json.

**Manejo de Errores:** Se utilizan bloques try y except para manejar posibles errores al abrir y escribir en el archivo.

**modificar\_producto(productos):** Permite modificar los detalles de un producto específico en una lista de productos, y luego actualizar el archivo JSON correspondiente.

**Mostrar productos**: Se llama a la función mostrar\_info\_productos() para mostrar la lista actual de productos.

**Entrada del usuario**: Se solicita al usuario que ingrese el id del producto que desea modificar.

**Búsqueda del producto**: Se recorre la lista de productos para encontrar el producto correspondiente al id ingresado.

* Si el producto no se encuentra, se le pide al usuario que ingrese otro id.

**Selección de campo a modificar**: Se ofrece un menú al usuario para seleccionar qué campo del producto desea modificar (nombre, marca, precio, ubicación, stock, promoción).

* Se actualizan los valores según la elección del usuario.

**Manejo de promociones**: Si se selecciona modificar la promoción, se permite al usuario ingresar detalles sobre la nueva promoción.

**Finalización**: Si el usuario elige finalizar (opción 7), la lista de productos se convierte a formato JSON y se guarda en el archivo productos.json.

**Manejo de Errores**: Se utilizan bloques try y except para manejar posibles errores al abrir y escribir en el archivo.

**mostrar\_info\_productos():** Se encarga de mostrar de manera estructurada la información de los productos almacenados en un archivo JSON.

**Leer productos**: Se llama a la función leer\_archivo() para obtener la lista de productos del archivo.

**Obtener las claves**: Se extraen las claves (nombres de las columnas) del primer elemento de la lista de productos. Esto asume que todos los productos tienen las mismas claves.

**Calcular el ancho de las columnas**: Se crea un diccionario anchuras que almacena el ancho necesario para cada columna, comenzando con el ancho del nombre de la clave.

* Luego, se recorre cada fila de productos para actualizar el ancho de cada columna en función de la longitud del contenido.

**Imprimir la cabecera**: Se crea una cadena de caracteres que representa la cabecera de la tabla, utilizando ljust() para alinear el texto.

* Se imprime una línea separadora debajo de la cabecera, basada en el ancho de cada columna.

**Imprimir las filas de datos**: Se recorren los productos y se imprimen cada uno de ellos en una línea, asegurando que los datos estén alineados correctamente.

**menu\_caja():**

**Inicialización**:

* Se lee el archivo de ventas (leer\_archivo\_ventas()), que contiene las ventas anteriores.
* Se obtiene la fecha actual para registrar las nuevas ventas.

**Menú principal**:

* Muestra un menú que ofrece dos opciones: "Ingresar productos" o "Volver".
* Si se elige "Ingresar productos", continúa con el proceso de venta. Si se elige "Volver", la función termina.

**Proceso de venta:**

* Se lee el archivo de productos (leer\_archivo()), que contiene los productos disponibles.
* El sistema muestra los productos disponibles y permite al usuario buscar uno (busqueda\_producto()).
* Se ingresa la cantidad a comprar y se verifica si hay suficiente stock.
* Calcula el importe total de la compra, aplicando descuentos o promociones si corresponde (obtener\_importe()).

**Actualización de ventas y stock:**

* La venta se guarda en el archivo de ventas, actualizando el stock del producto comprado.
* El archivo de productos también se actualiza con el nuevo stock.

**Agregar más productos:**

* El usuario puede elegir si quiere agregar más productos a la misma venta o terminar la compra.

**Mostrar resumen de la compra:**

* Se muestra un resumen de los productos comprados, con su precio, cantidad y el importe total.

**Método de pago:**

* Se pide al cliente elegir un método de pago (efectivo o tarjeta).
* Si paga en efectivo, se calcula y muestra el vuelto, si es necesario.

**Cierre:**

* El ciclo se repite si el usuario decide agregar más productos, o termina si elige volver.

**obtener\_importe(importe, promocion, cantidad):** Calcula el total a pagar dependiendo de la promoción y la cantidad de productos.

**Tres casos de promociones**:

* **Caso 1: "MxN" (ejemplo "2x3")**:
  + Compras N unidades y pagas por M de ellas.
  + La función cuenta cuántos grupos de N hay en la cantidad comprada, aplica la promoción en esos grupos, y para las unidades que sobran cobra el precio normal.
* **Caso 2: "325" (ejemplo "32% de descuento en la 5ta unidad")**:
  + Da un descuento solo en una unidad específica (como la 5ta).
  + Si compras esa cantidad o más, aplica el descuento en esa unidad específica, y el resto se cobra al precio normal.
* **Caso 3: "35" (ejemplo "35% de descuento en todas las unidades")**:
  + Aplica un descuento a todas las unidades compradas.

**Sin promoción**: Si no hay promoción, multiplica directamente el precio por la cantidad de productos.

**formato\_fecha():** Convierte una fecha en un formato de texto más fácil de leer, como "25 de Septiembre del 2024".

**Define los meses**: Crea una tupla con los nombres de los 12 meses (Enero, Febrero, etc.).

**Obtiene el día**: Usa fecha.day para extraer el número del día de la fecha.

**Obtiene el mes**: Con fecha.month, saca el número del mes y lo usa para buscar el nombre en la lista de meses. Como la lista empieza en 0, resta 1 para obtener el mes correcto.

**Obtiene el año**: Usa fecha.year para obtener el año.

**Formato de la fecha**:

Usa format() para combinar día, mes y año en una cadena que tiene el formato "día de mes del año".

**Retorno**: La función devuelve el texto con la fecha formateada.

**menu\_salir():** Se encarga de mostrar un banner o encabezado en forma de arte ASCII en la consola.

* **Definición del banner**: La función define una cadena de texto multilínea que contiene el arte ASCII.
* **Impresión del banner**: Utiliza la función print() para mostrar el banner en la consola.

**main ():** Es el punto de entrada principal del programa y se encarga de gestionar el flujo del mismo basado en las opciones del menú principal.

**Llamada al menú principal**: Se llama a la función menu\_principal() para mostrar las opciones del menú y obtener la elección del usuario. El valor retornado por esa función se almacena en opcion\_menu\_principal.

**Condicionales:**

**Opción 1:** Si el usuario elige 1, llama a la función menu\_info\_productos() para mostrar información de productos y luego vuelve a llamar a main() para mostrar el menú nuevamente.

**Opción 2:** Si elige 2, llama a menu\_abm\_productos() para gestionar productos y vuelve a llamar a main().

**Opción 3:** Si elige 3, llama a menu\_caja() para manejar ventas y también vuelve a llamar a main().

**Opción 4:** Si elige 4, simplemente imprime una línea en blanco.

**Opción no válida:** Si la opción no es ninguna de las anteriores, llama a menu\_salir() para manejar la salida del programa.