UNIVERSIDAD AMERICANA

Facultad de Ingeniería y Arquitectura

Introducción a la programación - Grupo 3

Ingeniería en Sistemas de Información

Autores:

Alexa Sofia Loaisiga Torrez

Chelsea Yosmara Quintanilla Blandón

Fabiola Sarai Lanuza Paz

Luis Lenin Aguirre Vilchez

Docente:

MSc. Silvia Gigdalia Ticay López

4 de julio de 2025

Índice

Introducción.

En la actualidad, el avance tecnológico y la digitalización de procesos representan factores clave para el fortalecimiento y competitividad de las pequeñas y medianas empresas. La automatización de tareas repetitivas y administrativas permite optimizar recursos, minimizar errores humanos y mejorar significativamente la eficiencia operativa. En este contexto, el presente proyecto surge ante la necesidad identificada en la empresa BLACEN, un negocio dedicado a la venta de repuestos y accesorios, que no cuenta con un sistema automatizado para llevar el control de su facturación e inventario. Esta limitación repercute directamente en la eficiencia interna y en la calidad del servicio al cliente.

El proyecto se desarrolla en un contexto tanto tecnológico como ambiental, abordando las múltiples limitaciones que implica el uso de un sistema de facturación físico en la gestión empresarial. Este tipo de sistema resulta poco práctico y obsoleto para las demandas actuales, ya que el proceso manual tiende a ser más lento, propenso a errores y requiere mayor esfuerzo operativo. La revisión, organización y análisis de datos se vuelven tareas complejas y tediosas al depender exclusivamente de documentos impresos, lo que complica la realización de auditorías, el seguimiento de movimientos históricos y la toma de decisiones basadas en información precisa y actualizada. Además, la introducción manual de datos aumenta considerablemente el margen de error, facilitando la aparición de problemas como cálculos incorrectos, fechas mal registradas, pérdidas de información o duplicación de registros. Desde una perspectiva ambiental, el uso excesivo de papel y recursos impresos también representa un impacto negativo, lo que refuerza la necesidad de migrar hacia soluciones digitales más sostenibles. En conjunto,

estas deficiencias evidencian la importancia de modernizar los procesos internos mediante la implementación de herramientas informáticas adaptadas a las necesidades reales del negocio.

Con el objetivo de resolver esta problemática, se desarrolló un programa informático de consola utilizando Python a través del IDE Visual Studio Code, precedido por la planificación y lógica en el entorno PSeInt. La elección de estas herramientas responde a su facilidad de aprendizaje, flexibilidad y amplia aplicabilidad en proyectos educativos y reales. El sistema diseñado permite registrar ventas, generar facturas además de llevar el control de productos en inventario, todo desde una interfaz sencilla. Aunque de carácter básico, este sistema representa un paso significativo hacia la digitalización de los procesos internos de la empresa.

Definición y alcance del caso de estudio:

Este caso de estudio se centra en la empresa BLACEN, un negocio dedicado a la comercialización de repuestos y accesorios para motos. La empresa enfrenta una limitación significativa en sus procesos administrativos, ya que actualmente no cuenta con un sistema automatizado para la gestión de facturación, control de inventario ni registro de ventas. Estas actividades se realizan de forma manual, lo que incrementa posibilidades de errores de cálculos, pérdida de información y retrasos en la atención al cliente. Esta situación afecta directamente la eficiencia operativa del negocio y su capacidad de responder de forma ágil a la demanda del mercado.

Con base a esta problemática, se ha desarrollado una solución informática en lenguaje Python, implementada a través de una interfaz de consola, cuyo propósito es automatizar las funciones clave relacionadas con la gestión comercial de la empresa. Previo a su programación se utilizó el entorno PSeInt para estructurar y organizar la lógica del sistema. Por tanto, el caso de estudio se enfoca en analizar la necesidad tecnológica de BLACEN y como herramienta sencilla pero funcional puede contribuir a mejorar su rendimiento administrativo y comercial.

El alcance del caso de estudio abarca las siguientes funcionalidades en el sistema: Registro y control de inventario, generación de facturas, gestión de ventas y una interfaz en consola. Este proyecto no contempla el desarrollo de una interfaz gráfica, ni la integración con bases de datos externas o servicios en la nube. Su propósito es mostrar la viabilidad de automatizar procesos clave en una pequeña empresa utilizando herramientas accesibles y programación estructurada. El sistema está orientado a un entorno local, monousuario y de uso interno para el personal administrativo de BLACEN.

Actividades de la práctica de familiarización:

Actividad 1: Descripción del problema/necesidad o caso de estudio

La empresa BLACEN a pesar del crecimiento sostenido que ha tenido en los últimos años, enfrenta una limitación considerable en sus procesos administrativos, ya que utiliza métodos manuales para la gestión de facturación, ventas e inventario. Esta situación se traduce en ineficiencias, pérdida de información, errores en el registro y una atención al cliente más lenta, lo cual afecta su competitividad en el mercado actual.

Actualmente, la facturación se realiza mediante máquinas pequeñas de impresión con papel térmico. Aunque esto agiliza la impresión, el proceso sigue siendo manual y propenso a errores, dado que los cálculos de totales, impuestos y descuentos se hacen fuera del sistema o con ayuda de calculadoras, lo que implica un riesgo significativo de equivocaciones humanas, especialmente en momentos de alta demanda. Además, el registro del inventario se lleva en una libreta que no siempre está actualizada, generando confusiones sobre la existencia real de productos. Estas limitaciones no solo obstaculizan el trabajo del personal, sino que también reducen la satisfacción del cliente y el control del negocio.

Para comprender con mayor profundidad la magnitud del problema, se llevó a cabo una investigación de campo que incluyó entrevistas, encuestas y observación directa. Se realizó una entrevista presencial con el señor **Juan Carlos Rodríguez**, **encargado general del negocio**, quien detalló el funcionamiento actual del sistema y las principales dificultades que enfrentan. Entre sus declaraciones más relevantes mencionó que cuando hay varios clientes esperando, se

vuelve estresante porque debe revisar los precios, llenar la factura, sacar la cuenta y escribir todo sin equivocarse; un error puede hacer perder tiempo y hasta dinero.

La entrevista permitió identificar otros aspectos importantes, como la falta de registros históricos organizados, la imposibilidad de generar reportes y el desgaste del personal al repetir tareas rutinarias que podrían ser automatizadas. También señaló que no cuentan con un respaldo digital de la información, por lo que ante una pérdida o daño de los documentos físicos no hay forma rápida de recuperar los datos.

La observación directa se llevó a cabo durante dos jornadas laborales completas, registrando en una bitácora las acciones del proceso de facturación e inventario. Se identificó que el tiempo promedio para atender a un cliente era de doce a dieciocho minutos, y que en momentos de mayor demanda las colas superaban los cinco clientes. Se detectaron errores en el cálculo de totales en al menos el treinta por ciento de las transacciones observadas, además de retrasos generados por tener que consultar la libreta del inventario para verificar existencias.

También se diseñó y aplicó una encuesta digital al personal de ventas, enfocada en medir su percepción sobre el sistema actual y su disposición hacia una solución automatizada. Los resultados mostraron que el ochenta por ciento considera que el sistema manual es poco eficiente, mientras que el cien por ciento está dispuesto a utilizar un sistema automatizado, siempre que sea fácil de usar y agilice su trabajo.

La implementación de un sistema automatizado representa una oportunidad para mejorar significativamente la gestión interna de BLACEN, permitiendo disminuir errores, optimizar tiempos de atención y conservar de forma segura y organizada la información relevante para la empresa. Esta mejora en los procesos contribuirá a fortalecer la competitividad y brindar un mejor servicio al cliente, aspectos fundamentales para el crecimiento sostenible del negocio.

Actividad 2: Análisis del problema

Objetivo general:

Desarrollar un sistema de gestión automatizado que permita optimizar los procesos de facturación, inventario y atención al cliente en la empresa BLACEN, como solución tecnológica accesible y funcional para mejorar su eficiencia operativa y modernización administrativa.

Objetivos específicos:

1. Recopilar datos importantes a través de una investigación de campo que incluya entrevistas, encuestas y observación directa en la empresa BLACEN, a fin de determinar los requerimientos más importantes del problema planteado.

2. Diseñar el algoritmo en el entorno PSeInt con el cual se presentará la lógica del "Sistema BLACEN", con una estructura sencilla y fácil de comprender.

3. Incorporar módulos para almacenar los datos de los clientes habituales, inventario, registro de facturas y credenciales de ingreso al sistema, utilizando archivos .txt y .json, a fin de llevar un control seguro de los procesos administrativos de la empresa BLACEN.

4. Crear una solución en consola que sea intuitiva y funcional, adaptable a usuarios con conocimientos básicos en informática, priorizando la usabilidad y claridad del sistema, desarrollada en lenguaje de programación Python a través del IDE Visual Studio Code.

5. Brindar capacitación al personal de la empresa BLACEN mediante charlas y un pequeño manual de instrucciones, para que aprendan a usar el sistema de facturación, registro y control de inventario. Esto ayudará a modernizar y hacer más eficientes sus procesos administrativos y operativos.

Respuestas a preguntas que llevan a definir bien el problema presentado.

¿Qué entradas se requieren? (tipo de datos con los cuales se trabaja y cantidad)

El sistema solicita primero el nombre de usuario y la contraseña para iniciar sesión, ambos como cadenas de texto. Luego, al realizar una venta, se ingresan el nombre del cliente, el RUC (opcional) y la fecha, también en formato texto. Para seleccionar productos, se pide el número del producto (entero) y la cantidad deseada (entero). Si se aplica un descuento, se debe ingresar el porcentaje en formato decimal.

Al agregar nuevos productos al inventario, el sistema solicita el nombre del producto, su categoría (texto), la cantidad (entero) y el precio (decimal). La cantidad de entradas puede variar según las operaciones que realice el usuario, pero todas se manejan entre cadenas, enteros y decimales. Estas entradas permiten controlar ventas, clientes e inventario de manera completa.

¿Cuál es la salida deseada? (tipo de datos de los resultados y cantidad)

La salida principal es la factura generada con datos como nombre del cliente, fecha, productos vendidos, cantidades, precios, subtotal, IVA, descuento y total a pagar. Estos resultados combinan texto y decimales, y se muestran en consola y se guardan en el archivo factura.txt.

También se actualiza el inventario y se muestra en pantalla en forma de tabla, utilizando texto, enteros y decimales. Además, se guarda un registro de los clientes atendidos en el archivo 'clientes_frecuentes.txt'. Estas salidas brindan control sobre ventas e inventario de forma clara y organizada.

¿Qué método produce la salida deseada?

La salida deseada del programa es la factura que se genera cada vez que se realiza una compra. Esta factura incluye el nombre del cliente, la fecha, los productos que se compraron, la cantidad, los precios, el subtotal, el IVA, el descuento si es que se aplica, y el total a pagar. Esta información se muestra en pantalla y se guarda en un archivo llamado 'factura.txt'.

Otra salida importante es el inventario actualizado, que se presenta en forma de tabla cada vez que se realiza una compra o cuando el usuario lo consulta desde el menú. En esta tabla se presentan los productos disponibles, su categoría, la cantidad que queda y el precio. todos estos datos también se guardan en un archivo llamado "inventario json".

Además, el programa guarda un registro de los clientes frecuentes en un archivo llamado “clientes_frecuentes.txt”, donde se almacena el nombre del cliente y su RUC si lo proporciona.

Los tipos de datos que se muestran en estas salidas son texto para los nombres, categorías y mensajes; números enteros para las cantidades; y decimales para los precios, el subtotal, el IVA, el descuento y el total. La cantidad de salidas dependerá del uso del programa, ya que se genera una factura por cada venta, se muestra el inventario cada vez que se consulta, y se registra un cliente cada vez que se atiende.

Requerimientos funcionales del SISTEMA BLACEN:

1. **Inicio de sesión seguro:**

   El sistema debe permitir el acceso solo a usuarios autorizados, con un máximo de 3 intentos.

2. **Gestión de inventario:**

- El sistema debe cargar automáticamente el inventario desde un archivo JSON al iniciar.
- Debe permitir al usuario agregar nuevos productos al inventario.
- Debe permitir eliminar productos existentes.
- Debe mostrar una vista detallada del inventario actual (número, nombre, categoría, cantidad y precio).

3. **Registro de ventas:**

- El sistema debe permitir realizar ventas seleccionando productos desde el inventario.
- Debe verificar la disponibilidad del stock antes de completar la venta.
- Al vender, debe descontar automáticamente la cantidad vendida del inventario.

4. **Generación de factura:**

- El sistema debe generar una factura con detalles de cliente, productos vendidos, cantidades, precios, subtotal, IVA (15%), descuentos y total.
- Debe guardar cada factura en un archivo de texto plano.

5. **Aplicación de descuentos:**

- El sistema debe permitir aplicar un porcentaje de descuento si el usuario lo indica.

6. **Registro de clientes frecuentes:**

- El sistema debe guardar los clientes con nombre y RUC, evitando duplicados.
- Debe permitir visualizar la lista de clientes frecuentes.

7. **Persistencia de datos:**

- Los cambios realizados en inventario, facturas y clientes deben guardarse automáticamente en sus respectivos archivos (.json, .txt).

Requerimientos NO funcionales del SISTEMA BLACEN:

1. **Usabilidad**

   El sistema debe tener una interfaz por consola clara, ordenada y fácil de usar para cualquier usuario sin conocimientos técnicos avanzados.

2. **Portabilidad**

   Debe funcionar en cualquier sistema operativo que soporte Python (Windows, Linux, macOS).

3. **Seguridad básica**

   La contraseña no debe mostrarse en pantalla al momento de ingresarla (uso de pwinput).

4. **Escalabilidad básica**

   El sistema debe permitir añadir más productos, usuarios o clientes sin necesidad de modificar la estructura del programa.

5. **Mantenimiento**

   El código debe estar modularizado, documentado con comentarios internos y dividido por funcionalidades para facilitar futuras mejoras o correcciones.

6. **Almacenamiento local**

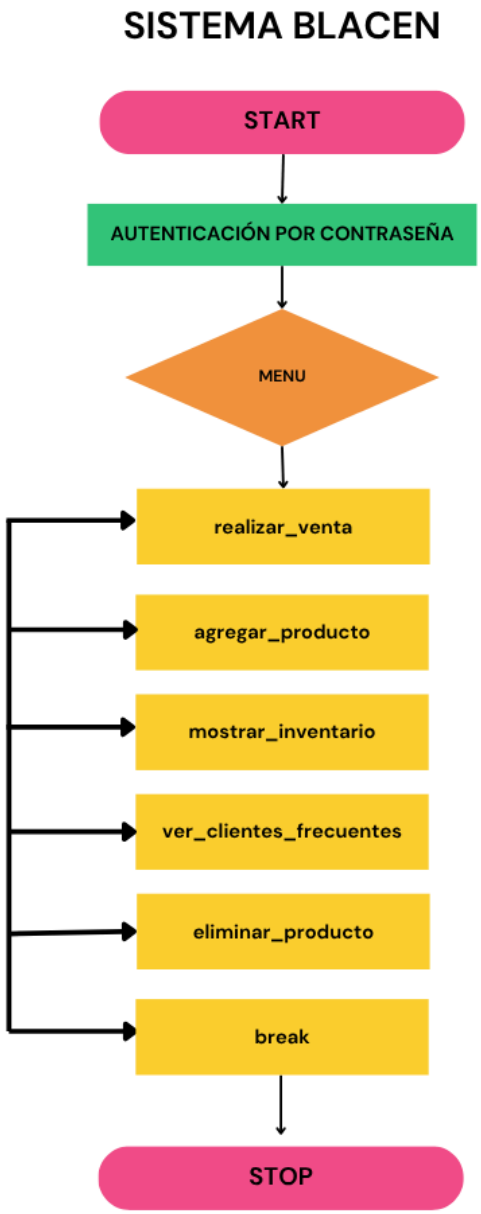
   Todos los datos deben almacenarse en archivos locales (.json y .txt), sin necesidad de conexión a internet o base de datos externa.

7. **Tiempo de respuesta:**

   El sistema debe responder inmediatamente a cada entrada del usuario (sin demoras perceptibles).

DIAGRAMA DE ESTRUCTURA DEL PROGRAMA.

- Presentamos el diagrama guía del Sistema BLACEN.



Actividad 3: Diseño del algoritmo

El pseudocódigo de nuestro proyecto fue elaborado con PSeInt, el cual es una herramienta educativa diseñada para ayudar a los estudiantes a aprender lógica de programación utilizando pseudocódigo en español. Sus principales funcionalidades incluyen un editor sencillo para escribir algoritmos, la posibilidad de ejecutar y simular los programas paso a paso, así como herramientas de depuración que permiten observar el valor de las variables y detectar errores de forma didáctica. Además, facilita la comprensión de estructuras básicas como condicionales, bucles y procedimientos, sin necesidad de conocer la sintaxis de un lenguaje de programación real.

Descripción técnica:

El pseudocódigo al ser una parte fundamental de la organización para formar nuestro sistema, a continuación se explicarán las funciones principales del sistema para comprender mejor el algoritmo.

VALIDACIÓN DE USUARIO.

Se utiliza estos condicionales para hacer la toma de decisiones como:

SI ususarioIngresado=ususarioCorrecto y ContraseñaIngresada=contraseña correcta Entonces

para verificar si las credenciales son correctas **SI** la condición es verdadera se concede el acceso pero **SINO** la condición es falsa se incrementa un contador de intentos.

LÍMITE DE INTENTOS: Es la parte de la repetición para ingresar la contraseña del usuario con la condición Hasta que (ususarioIngresado=ususarioCorrecto y ContraseñaIngresada=contraseña

correcta) O intentos=3. este bucle indica cuando finaliza si es acceso es exitoso o si se alcanzan los intentos fallidos.

SI intentos=3 Entonces: Tras el bucle, se va a evaluar si la salida fue por exceder los intentos. Si esa condición es verdadera, se negara el acceso definidamente.

SELECCIÓN DE OPCIONES:

Mediante este módulo se emplea la estructura Según opción Hacer para dirigir el flujo del programa en función de venta mostrando las 4 opciones para realizar el algoritmo De otro Modo si la opción ingresada no coincide con ninguna de las anteriores, se informa al usuario que la opción es inválida.

SELECCIÓN DE PRODUCTO Y VALIDACIÓN:

- SI indiceProducto>=1 Y indiceProducto<=totalProducto

  ENTONCES:Verifica que el número de producto ingresado por el usuario esté dentro del rango válido del inventario, SINO:Si la condición es falsa, informa que el número es inválido.

VERIFICACIÓN DE STOCK:

Dentro de la selección de producto válida, se utiliza Si cantidades [indiceProducto]>=cantidadVenta Entonces para asegurar que hay suficiente cantidad del producto en el inventario para la venta.

Si la condición es verdadera, se procede con la venta (actualización de subtotal, inventario, y registro en factura), SiNO: Si la condición es falsa, se notifica que no hay suficiente inventario.

ADICIÓN DE MULTIPLES PRODUCTOS A LA FACTURA:

- Un bucle Repetir Hasta Que Mayúsculas (respuesta)= NO permite al usuario añadir tantos productos como desee a la misma factura. La condición Mayusculas (respuesta)= NO determina cuándo finalizar la adición de ítems.

1) SISTEMA DE SEGURIDAD

```
Definir usuarioCorrecto, contrasenaCorrecta Como Cadena
usuarioCorrecto ← "admin"
contrasenaCorrecta ← "BLACEN"
Repetir
    Escribir '======= ACCESO AL SISTEMA BLACEN ======='
    Escribir "Ingrese usuario:"
    Leer usuarioIngresado
    Escribir 'Ingrese contrase?a:'
    Leer contrasenaIngresada
    Si usuarioIngresado=usuarioCorrecto Y contrasenaIngresada=contrasenaCorrecta Entonces
        Escribir 'Acceso concedido. Bienvenido ', usuarioCorrecto, '.'
    SiNo
        intentos ← intentos+1
        Escribir 'Usuario o contrase?a incorrectos. Intento ', intentos, ' de 3.'
    FinSi
Hasta Que (usuarioIngresado=usuarioCorrecto Y contrasenaIngresada=contrasenaCorrecta) O intentos=3
Si intentos=3 Entonces
    Escribir 'Demasiados intentos fallidos. Acceso denegado.'
FinSi
```

DESCRIPCIÓN: Mediante esta primera parte del algoritmo permite controlar el acceso principal al sistema para solicitar el acceso inicial al sistema permitiendo solicitar el nombre del usuario y contraseña. Solo permite 3 intentos antes de denegar el acceso.

2) DECLARACIÓN DE DATOS Y VARIABLES

```
    // Declaraciones
    Dimensionar productos(100)
    Dimensionar categorias(100)
    Dimensionar cantidades(100)
    Dimensionar precios(100)
    Dimensionar facturaProducto(50)
    Dimensionar facturaCantidad(50)
    Dimensionar facturaPrecio(50)
    Dimensionar facturaTotal(50)
    // Definici?n de variables generales
    Definir totalProductos, opcion, i, indiceProducto, cantidadVenta, productoEncontrado Como Entero
    Definir subtotal, iva, descuento, granTotal, precioUnitario, porcentajeDescuento Como Real
    Definir numFacturaItems Como Entero
    Definir cliente, ruc, fecha, respuesta Como Cadena
    Definir prodNuevo, catNueva Como Cadena
    Definir cantNueva Como Entero
    Definir precioNuevo Como Real
    // Inicializar productos (13 productos base)
    totalProductos ← 13
    productos[1] ← 'Aceite Castrol'
    categorias[1] ← 'Aceites y Lubricantes'
    cantidades[1] ← 10
    precios[1] ← 400
    productos[2] ← 'Aceite Axiom'
    categorias[2] ← 'Aceites y Lubricantes'
    cantidades[2] ← 10
    precios[2] ← 450
    productos[3] ← 'Neum?tico 3.25-17 GB Boy'
    categorias[3] ← 'Neum?ticos y Llantas'
    cantidades[3] ← 10
```

DESCRIPCIÓN: Mediante la segunda parte del pseudocódigo se encarga de la asignación de todas las variables y cada parte de la estructura de datos que son necesarias para el funcionamiento del sistema como: Los productos, precios y categorías y así poder preparar el inventario inicial.

3) MENÚ PRINCIPAL

```
precios[15] ← 600
// Menu principal
Repetir
    Escribir '========= SISTEMA BLACEN ========='
    Escribir '1. Realizar venta'
    Escribir '2. Agregar producto nuevo'
    Escribir '3. Ver inventario'
    Escribir '4. Salir'
    Escribir 'Seleccione una opcion:'
    Leer opcion
    Segun opcion Hacer
```

DESCRIPCIÓN: Este módulo es lo principal del sistema ya que muestra el menú de opciones para el usuario al escoger alguna de estas opciones y así realizar el objetivo del algoritmo.

A) REALIZAR VENTA

```
1:
    subtotal ← 0
    numFacturaItems ← 0
    Escribir 'Ingrese nombre del cliente:'
    Leer cliente
    Escribir 'Ingrese RUC (opcional):'
    Leer ruc
    Escribir 'Ingrese fecha:'
    Leer fecha
    Repetir
        Escribir '------- INVENTARIO BLACEN -------'
        Para i←1 Hasta totalProductos Con Paso 1 Hacer
            Escribir i, '. ', productos[i], ' (', categorias[i], ') - Cantidad: ', cantidades[i], ' - Precio: C$', precios[i]
        FinPara
        Escribir 'Ingrese el numero del producto a vender:'
        Leer indiceProducto
        Si indiceProducto≥1 Y indiceProducto≤totalProductos Entonces
            Escribir 'Cantidad disponible: ', cantidades[indiceProducto]
            Escribir 'Ingrese cantidad a vender:'
            Leer cantidadVenta
            Si cantidades[indiceProducto]≥cantidadVenta Entonces
                precioUnitario ← precios[indiceProducto]
                subtotal ← subtotal+(precioUnitario*cantidadVenta)
                cantidades[indiceProducto] ← cantidades[indiceProducto]-cantidadVenta
                numFacturaItems ← numFacturaItems+1
                facturaProducto[numFacturaItems] ← indiceProducto
            SiNo
                Escribir 'No hay suficiente inventario.'
            FinSi
        SiNo
            Escribir 'Numero invalido.'
        FinSi
        Escribir '?Desea agregar otro producto a la factura? (SI/NO):'
        Leer respuesta
    Hasta Que Mayusculas(respuesta)='NO'
    iva ← subtotal*0.15
    descuento ← 0
    Escribir '?Aplicar descuento por temporada? (SI/NO):'
    Leer respuesta
    Si Mayusculas(respuesta)='SI' Entonces
        Escribir 'Ingrese porcentaje de descuento:'
        Leer porcentajeDescuento
        descuento ← subtotal*(porcentajeDescuento/100)
    FinSi
    granTotal ← subtotal+iva-descuento
    Escribir '========== FACTURA BLACEN =========='
    Escribir 'Cliente: ', cliente
    Si ruc≠'' Entonces
        Escribir 'RUC: ', ruc
    FinSi
    Escribir 'Fecha: ', fecha
    Escribir '      Detalle de productos:'
    Escribir 'Producto | Cant | Precio | Total'
    Para i←1 Hasta numFacturaItems Con Paso 1 Hacer
        indiceProducto ← facturaProducto[i]
        Escribir productos[indiceProducto], ' | ', facturaCantidad[i], ' | C$', facturaPrecio[i], ' | C$', facturaTotal[i]
    FinPara
    Escribir 'Subtotal: C$', subtotal
    Escribir 'IVA 15%: C$', iva
```

DESCRIPCIÓN: Para nuestro primer inciso del menú explica que realizará una venta que ahí es donde se agrega el nombre del cliente el producto, la cantidad de productos que llevara el cliente teniendo en cuenta si está o no en inventario ya que tiene registrado cuánto hay de cada producto

gracias al inventario y el precio que aparte de añadir el total se agrega el IVA, utilizando condicional (SI).

B) AGREGAR PRODUCTO NUEVO

```
Escribir '====================================='
2:
totalProductos ← totalProductos+1
Escribir 'Ingrese nombre del nuevo producto:'
Leer prodNuevo
productos[totalProductos] ← prodNuevo
Escribir 'Ingrese categor?a:'
Leer catNueva
categorias[totalProductos] ← catNueva
Escribir 'Ingrese cantidad:'
Leer cantNueva
cantidades[totalProductos] ← cantNueva
Escribir 'Ingrese precio:'
Leer precioNuevo
precios[totalProductos] ← precioNuevo
Escribir 'Producto agregado correctamente.'
3:
```

DESCRIPCIÓN: Para el segundo módulo sirve para añadir más productos al inventario y así obtener más productos y variedad al inventario del sistema.

C) VER INVENTARIO

```
Escribir 'Producto agregado correctamente.'
3:
Escribir '------- INVENTARIO ACTUAL -------'
Para i←1 Hasta totalProductos Con Paso 1 Hacer
    Escribir i, '. ', productos[i], ' - ', categorias[i], ' - Cantidad: ', cantidades[i], ' - Precio: C$', precios[i]
FinPara
```

DESCRIPCIÓN: En este tercer módulo tenemos inventario esta parte es una parte crucial para mantener un orden y un control sobre la cantidad y variedad de productos que maneja nuestro sistema para así ahorrar tiempo de búsqueda y de trabajo.

D) SALIR

```
192         4:
193             Escribir 'Gracias por usar el sistema BLACEN. Hasta pronto.'
194         De Otro Modo:
195             Escribir 'Opci?n inv?lida. Intente de nuevo.'
196             FinSeg?n
197         Hasta Que opcion=4
198     FinAlgoritmo
199
```

Descripción: Para el cuarto módulo es la última parte de nuestro sistema para dar una descripción para anunciar que nuestro sistema llegó a su fin.

Nota: El pseudocódigo solamente presenta 4 opciones en el menú debido a que la herramienta PSeInt no cuenta con la capacidad de interactuar directamente con archivos, se le agregó al código Python 2 opciones más que interactúan con archivos, como "ver_clientes_frecuentes" y "eliminar_producto"

Actividad 4: Codificación, ejecución, verificación y depuración

El sistema BLACEN se desarrolló en Python, este es un lenguaje de programación informático que se utiliza a menudo para crear sitios web y software, automatizar tareas y realizar análisis de datos. Python es un lenguaje de propósito general, lo que significa que se puede utilizar para crear una variedad de programas diferentes y no está especializado en ningún problema específico. Esta versatilidad, junto con su facilidad para los principiantes, lo ha convertido en uno de los lenguajes de programación más utilizados en la actualidad.

Algunas de sus ventajas son:

- Tiene una sintaxis sencilla que imita el lenguaje natural, por lo que es más fácil de leer y entender. Esto hace la construcción y mejora de los proyectos más rápida.
- Es versátil. Python puede utilizarse para muchas tareas diferentes, desde el desarrollo web hasta el aprendizaje automático.
- Es fácil de usar para los principiantes, por lo que es muy popular entre los programadores principiantes.
- Es de código abierto, lo que significa que su uso y distribución son gratuitos, incluso para fines comerciales.
- El archivo de módulos y bibliotecas de Python—conjuntos de código creados por otros usuarios para ampliar las capacidades de Python—es enorme y sigue creciendo.

Además se utilizó el IDE de programación llamado **Visual Studio Code** el cual es el editor de código fuente desarrollado por Microsoft, diseñado para ser ligero, rápido y altamente personalizable. A diferencia de otros editores de código, como los IDE completos (Entornos de Desarrollo Integrados), VS Code se enfoca en ofrecer una experiencia ágil para la escritura y edición de código, sin sacrificar características avanzadas.

Entre sus características clave, se incluyen la autocompletado inteligente mediante IntelliSense, la depuración integrada, y el control de versiones con Git, lo que convierte a VS Code en una opción versátil tanto para desarrolladores individuales como para equipos.

Uno de los mayores puntos a favor de Visual Studio Code es que es completamente gratuito y multiplataforma. Esto significa que se puede descargar y utilizar VS Code en Windows, macOS y Linux sin costo alguno. Esta accesibilidad ha facilitado su adopción tanto por parte de desarrolladores principiantes como por equipos profesionales que buscan una herramienta eficiente sin tener que invertir en costosas licencias de software. Además, al ser un proyecto de código abierto, la comunidad puede contribuir al desarrollo del editor y crear nuevas extensiones para mejorar su funcionalidad.

Para desarrollar el Sistema BLACEN adaptado a Python a través del IDE Visual Studio Code fue necesario utilizar muchas de las funcionalidades que el lenguaje de programación nos proporciona:

En primer lugar se utilizó **definición de funciones** las cuales consisten en:

- ❖ Definir un bloque de código reutilizable que se puede ejecutar muchas veces dentro de tu programa.

- ❖ La sintaxis de esta es:
  - La palabra clave def.
  - Un nombre de función
  - Paréntesis (), y dentro de los paréntesis los parámetros de entrada, aunque los parámetros de entrada sean opcionales.
  - Dos puntos :
  - Bloque de código para ejecutar.
  - Una sentencia de retorno (opcional).

```python
#Funcion para la validación del inicio de sesión, máximo 3 intentos
def login():
    print("=== INICIO DE SESIÓN SISTEMA BLACEN===")
    usuarios= cargar_usuarios()

    intentos = 3
    while intentos > 0:
        usuario = input("Usuario: ")
        contrasena_ingresada = pwinput.pwinput("Contraseña: ", mask='*')

        if usuario in usuarios and usuarios[usuario] == contrasena_ingresada:
            print("¡Acceso concedido!\n")
            return True
        else:
            print("Credenciales incorrectas.")
            intentos -= 1
    print("Demasiados intentos fallidos.")
    return False
```

Este es un ejemplo de definición de funciones aplicado a nuestro sistema BLACEN

Además se implementa en nuestro código el uso de los condicionales:

If, elif, else, estos consisten en:

- **Sentencia IF:** Si la condición que sigue a la palabra clave if se evalúa como verdadera, el bloque de código se ejecutará. Ten en cuenta que los paréntesis no se utilizan antes y después de la verificación de condición como en otros idiomas.
- **Sentencia ELSE:** Opcionalmente, puedes agregar una respuesta else que se ejecutará si la condición es false.
- **Sentencia ELIF:** Se pueden verificar varias condiciones al incluir una o más verificaciones elif después de su declaración if inicial. Se debe tener en cuenta que solo se ejecutará una condición**.**

Se implementa el uso de bucles:

Los bucles son otra herramienta para alterar el flujo normal de un programa. Nos permiten repetir una porción de código tantas veces como queramos. Python incluye únicamente dos tipos de bucle: *while* y *for*.

- Bucle *for*: Itera sobre una secuencia dada.
- Bucle *while*: se repite mientras se cumpla cierta condición booleana.
- Sentencia *break*: se utiliza para salir de un bucle *mientras*.

Usamos la función set() la cual es una colección no ordenada de elementos únicos e inmutables. Esto significa que un set no permite duplicados, y el orden de los elementos no está garantizado.

Usamos las excepciones *try* y except los cuales están basados en:

- El bloque *try* es el bloque con las sentencias que quieres ejecutar. Sin embargo, podrían llegar a haber errores de ejecución y el bloque se dejará de ejecutar.
- El bloque *excepto* se ejecutará cuando el bloque *try* falle debido a un error. Este bloque contiene sentencias que generalmente nos dan un contexto de lo que salió mal en el bloque *try*.

Métodos de cadena utilizados:

- El .lower() método se utiliza con frecuencia cuando se comparan dos cadenas para normalizar sus mayúsculas y minúsculas.
- El método .upper() convierte los caracteres de una cadena a mayúsculas.
- El método .split() se utiliza para listar cada palabra de una cadena.

Archivos utilizados:

- .txt: En Python, un archivo TXT es un archivo de texto plano que almacena datos como caracteres y cadenas. Se utiliza comúnmente para almacenar código fuente, datos estructurados en texto o cualquier tipo de información legible. Python proporciona funciones para leer y escribir en archivos TXT, permitiendo la manipulación de datos almacenados en ellos.
- .json: En Python, los archivos JSON (JavaScript Object Notation) se utilizan para almacenar datos estructurados en un formato ligero y legible por humanos. El módulo json de Python facilita la lectura y escritura de estos archivos, convirtiéndolos desde y hacia objetos Python como diccionarios y listas.

La implementación del sistema BLACEN en Python nos permitió aplicar de forma práctica y estructurada los fundamentos esenciales de la programación, como funciones, condicionales, bucles, manejo de excepciones y operaciones con cadenas. Estas herramientas no solo facilitaron la organización del código, sino que también nos brindaron una manera clara, flexible y robusta de resolver problemas reales, como el control de inventario y la gestión de ventas. Al desarrollar este sistema, no solo reforzamos nuestros conocimientos técnicos, sino que también comprobamos cómo la programación puede ofrecer soluciones eficientes y significativas en contextos empresariales.

Actividad 5: Documentación del proyecto

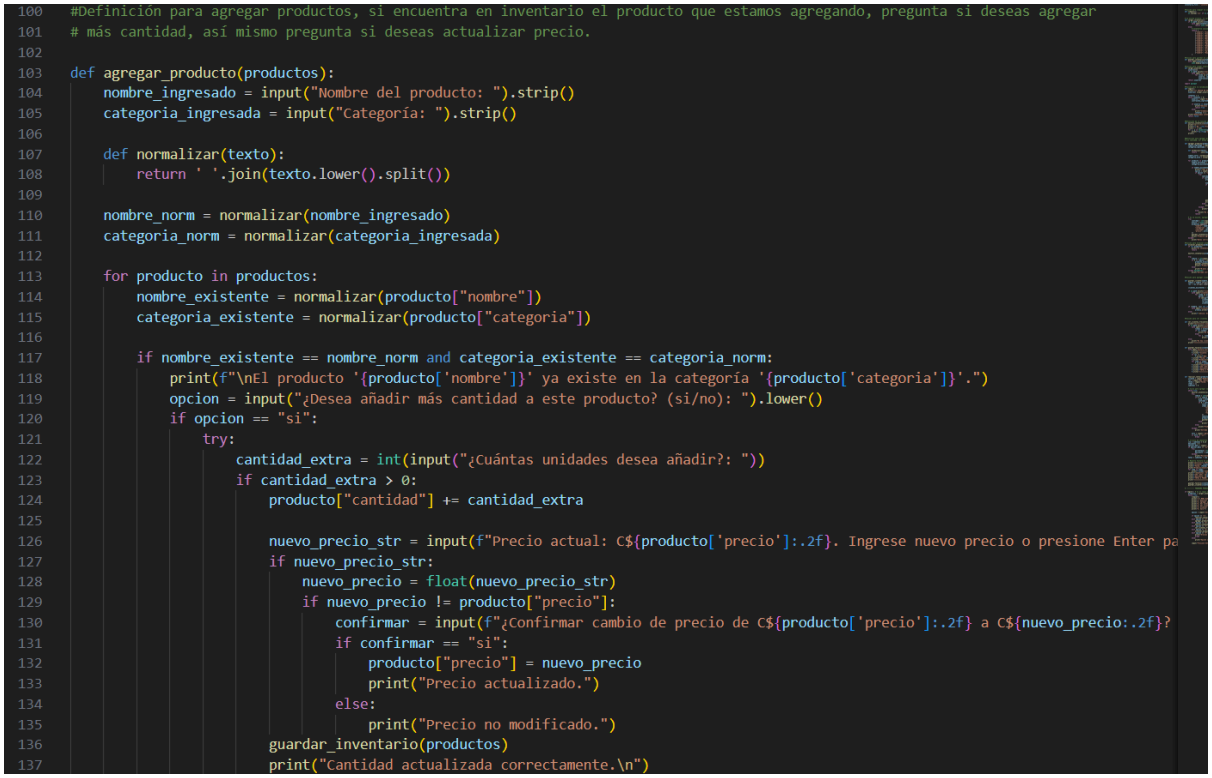
Se muestran a continuación algunas capturas de pantalla que contienen partes esenciales del código completo, a fin de describir lo que se realiza.

```python
#Funcion para la validación del inicio de sesión, máximo 3 intentos
def login():
    print("=== INICIO DE SESIÓN SISTEMA BLACEN===")
    usuarios= cargar_usuarios()

    intentos = 3
    while intentos > 0:
        usuario = input("Usuario: ")
        contrasena_ingresada = pwinput.pwinput("Contraseña: ", mask='*')

        if usuario in usuarios and usuarios[usuario] == contrasena_ingresada:
            print("¡Acceso concedido!\n")
            return True
        else:
            print("Credenciales incorrectas.")
            intentos -= 1
    print("Demasiados intentos fallidos.")
    return False

```

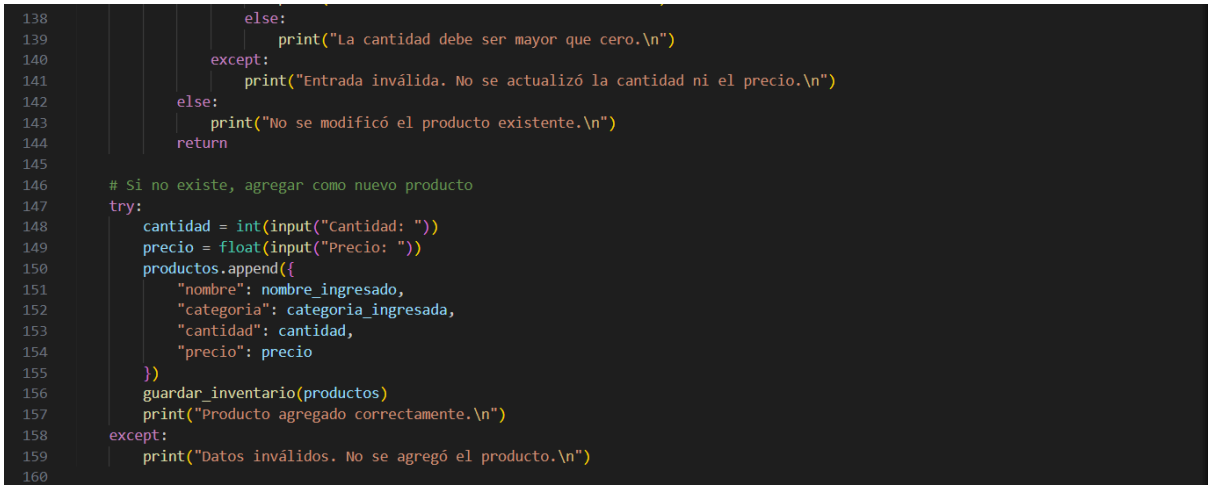
def login(): Permite el acceso al sistema solicitando usuario y contraseña. Tiene un máximo de 3 intentos y oculta la contraseña al escribirla. Se utiliza la librería de pwinput para ocultar la contraseña al ingresar.

```python

#Definición de la función para mostrar el inventario
def mostrar_inventario(productos):
    print("\n---------------------------------------------- INVENTARIO ACTUAL ----------------------------------------------------------")
    print(f"{'No.':<7}{'Producto':<40}{'Categoría':<50}{'Cant':<6}{'Precio (C$)':<10}")
    print("-" * 115)
    for i, p in enumerate(productos, 1):
        print(f"{i:<7}{p['nombre']:<40}{p['categoria']:<50}{p['cantidad']:<6}{p['precio']:<10.2f}")
    print()

```

def mostrar_inventario(productos): Muestra en pantalla una tabla organizada con todos los productos del inventario, incluyendo nombre, categoría, cantidad y precio.



```python
#Definición para agregar productos, si encuentra en inventario el producto que estamos agregando, pregunta si deseas agregar
# más cantidad, así mismo pregunta si deseas actualizar precio.

def agregar_producto(productos):
    nombre_ingresado = input("Nombre del producto: ").strip()
    categoria_ingresada = input("Categoría: ").strip()

    def normalizar(texto):
        return ' '.join(texto.lower().split())

    nombre_norm = normalizar(nombre_ingresado)
    categoria_norm = normalizar(categoria_ingresada)

    for producto in productos:
        nombre_existente = normalizar(producto["nombre"])
        categoria_existente = normalizar(producto["categoria"])

        if nombre_existente == nombre_norm and categoria_existente == categoria_norm:
            print(f"\nEl producto '{producto['nombre']}' ya existe en la categoría '{producto['categoria']}'.")
            opcion = input("¿Desea añadir más cantidad a este producto? (si/no): ").lower()
            if opcion == "si":
                try:
                    cantidad_extra = int(input("¿Cuántas unidades desea añadir?: "))
                    if cantidad_extra > 0:
                        producto["cantidad"] += cantidad_extra

                        nuevo_precio_str = input(f"Precio actual: C${producto['precio']:.2f}. Ingrese nuevo precio o presione Enter pa
                        if nuevo_precio_str:
                            nuevo_precio = float(nuevo_precio_str)
                            if nuevo_precio != producto["precio"]:
                                confirmar = input(f"¿Confirmar cambio de precio de C${producto['precio']:.2f} a C${nuevo_precio:.2f}?
                                if confirmar == "si":
                                    producto["precio"] = nuevo_precio
                                    print("Precio actualizado.")
                                else:
                                    print("Precio no modificado.")
                        guardar_inventario(productos)
                        print("Cantidad actualizada correctamente.\n")
```



```python
                    else:
                        print("La cantidad debe ser mayor que cero.\n")
                except:
                    print("Entrada inválida. No se actualizó la cantidad ni el precio.\n")
            else:
                print("No se modificó el producto existente.\n")
            return

    # Si no existe, agregar como nuevo producto
    try:
        cantidad = int(input("Cantidad: "))
        precio = float(input("Precio: "))
        productos.append({
            "nombre": nombre_ingresado,
            "categoria": categoria_ingresada,
            "cantidad": cantidad,
            "precio": precio
        })
        guardar_inventario(productos)
        print("Producto agregado correctamente.\n")
    except:
        print("Datos inválidos. No se agregó el producto.\n")

```

def agregar_producto(productos): Permite agregar un nuevo producto al inventario o actualizar la cantidad y/o precio de un producto existente. Esta definición de funciones contiene bucle *for*, condicionales *if, else*, además de las excepciones *try* y *except.* Además

```python
#Función para eliminar productos del inventario
def eliminar_producto(productos):
    if not productos:
        print("El inventario está vacío. No hay productos para eliminar.\n")
        return

    mostrar_inventario(productos)

    try:
        indice = int(input("Ingrese el número del producto que desea eliminar: ")) - 1
        if 0 <= indice < len(productos):
            producto_eliminado = productos.pop(indice)
            guardar_inventario(productos)
            print(f"Producto '{producto_eliminado['nombre']}' eliminado correctamente.\n")
        else:
            print("Número inválido. No se eliminó ningún producto.\n")
    except ValueError:
        print("Entrada inválida. Por favor ingrese un número válido.\n")

```

def eliminar_producto(productos): Permite eliminar un producto del inventario seleccionándolo por número. Actualiza el archivo *inventario.json* tras la eliminación.

```python
def realizar_venta(productos):
    cliente = input("Nombre del cliente: ")
    ruc = input("RUC (opcional): ")
    fecha = input("Fecha: ")
    items = []
    subtotal = 0

    # Ciclo para agregar varios productos
    while True:
        mostrar_inventario(productos)
        try:
            indice = int(input("Ingrese número del producto a vender: ")) - 1
            if 0 <= indice < len(productos):
                prod = productos[indice]
                cantidad = int(input(f"Ingrese cantidad a vender (Disponible: {prod['cantidad']}): "))
                if 0 < cantidad <= prod["cantidad"]:
                    total_item = cantidad * prod["precio"]
                    items.append({
                        "nombre": prod["nombre"],
                        "cantidad": cantidad,
                        "precio": prod["precio"],
                        "total": total_item
                    })
                    subtotal += total_item
                    productos[indice]["cantidad"] -= cantidad
                    guardar_inventario(productos)
                else:
                    print("Cantidad inválida.")
            else:
                print("Producto no encontrado.")
        except:
            print("Entrada inválida.")

        otro = input("¿Agregar otro producto? (si/no): ").lower()
        if otro != "si":
            break
```

def realizar_venta(productos): Gestiona una venta completa: selección de productos, verificación de stock, cálculo de subtotal, IVA y descuentos. Actualiza el inventario y genera una factura en pantalla y en archivo.

```python
#Función para agregar clientes a nuestros archivos especiales para ello: clientes_frecuentes.txt

def guardar_cliente(nombre, ruc):
    nombre = nombre.strip().lower()
    ruc = ruc.strip().lower()

    clientes_existentes = set()

    if os.path.exists("clientes_frecuentes.txt"):
        with open("clientes_frecuentes.txt", "r") as archivo:
            for linea in archivo:
                partes = linea.strip().split("-")
                if len(partes) >= 2:
                    cliente_guardado = partes[0].strip().lower()
                    ruc_guardado = partes[1].replace("RUC:", "").strip().lower()
                    clientes_existentes.add((cliente_guardado, ruc_guardado))

    if (nombre, ruc) not in clientes_existentes:
        with open("clientes_frecuentes.txt", "a") as archivo:
            archivo.write(f"{nombre.title()} - RUC: {ruc.upper()}\n")
    else:
        print(f"\nAviso: El cliente '{nombre.title()}' con RUC '{ruc.upper()}' ya está registrado como frecuente.\n")

```

def guardar_cliente(nombre, ruc): Agrega al cliente actual a un archivo de clientes frecuentes (clientes_frecuentes.txt), evitando duplicados. Normaliza los datos y verifica si el cliente ya está en el archivo antes de agregarlo. Esto ayuda a construir una base de clientes recurrentes para seguimiento o promociones.

```python
def ver_clientes_frecuentes():
    print("\n========== CLIENTES FRECUENTES ==========")
    if os.path.exists("clientes_frecuentes.txt"):
        with open("clientes_frecuentes.txt", "r") as archivo:
            lineas = archivo.readlines()
            if not lineas:
                print("No hay clientes frecuentes registrados.")
            else:
                for i, linea in enumerate(lineas, 1):
                    print(f"{i}. {linea.strip()}")
    else:
        print("No hay clientes frecuentes registrados.")
    print("=========================================\n")

```

def ver_clientes_frecuentes(): Lee el archivo clientes_frecuentes.txt y muestra en pantalla la lista ordenada de todos los clientes registrados como frecuentes. Permite al usuario visualizar qué clientes ya han realizado compras.

```
58  #Funcion para cargar usuarios
59  def cargar_usuarios():
60      usuarios={}
61      if os.path.exists("users.txt"):
62          with open("users.txt", "r") as archivo:
63              for linea in archivo:
64                  usuario, contrasena=linea.strip().split(",")
65                  usuarios[usuario] = contrasena
66      return usuarios
```

def cargar_usuarios(): Lee el archivo users.txt donde se almacenan usuarios válidos del sistema. Devuelve un diccionario donde las claves son nombres de usuario y los valores sus contraseñas. Este diccionario es utilizado por la función login() para validar credenciales.

```python
if login(): # Si el login es exitoso, entra al menú
    productos = cargar_inventario()
    while True:
        limpiar()
        print("===== MENÚ BLACEN =====")
        print("1. Realizar venta")
        print("2. Agregar producto nuevo")
        print("3. Ver inventario")
        print("4. Ver clientes frecuentes")
        print("5. Eliminar producto")
        print("6. Salir")

        opcion = input("Seleccione una opción: ")

        if opcion == "1":
            realizar_venta(productos)
        elif opcion == "2":
            agregar_producto(productos)
        elif opcion == "3":
            mostrar_inventario(productos)
        elif opcion == "4":
            ver_clientes_frecuentes()
        elif opcion == "5":
            eliminar_producto(productos)
        elif opcion == "6":
            print("Gracias por usar el sistema BLACEN.")
            break
        else:
            print("Opción inválida.\n")

        input("Presione Enter para continuar...")

```

Menú BLACEN: cuenta con 6 opciones de llamados a las funciones definidas previamente, las opciones 4 y 5 fue posible agregarlas en el código de Python gracias al acceso a archivos, caso contrario al pseudocódigo de PSeInt.

Conclusiones.

El sistema BLACEN fue desarrollado con el objetivo de automatizar y optimizar procesos internos en la empresa dedicada a la venta de repuestos y accesorios de motocicletas. A través del uso del lenguaje de programación Python, desarrollado en el IDE Visual Studio Code, complementado con la lógica previa en PSeInt, se logró construir una herramienta funcional que facilita el registro de ventas, la generación de facturas y la gestión del inventario. Este avance representa un paso importante hacia la digitalización de las operaciones de una pequeña empresa.

La implementación del sistema permitió aplicar una variedad de estructuras y conceptos fundamentales de programación como funciones, condicionales, bucles, manejo de archivos, validación de entradas y manejo de excepciones. Además, se integraron herramientas de persistencia de datos como archivos JSON y TXT para guardar inventario, clientes frecuentes y facturas, lo que contribuyó a mantener la trazabilidad y organización de la información.

Aunque se trata de una versión básica en consola, el sistema demostró ser funcional, flexible y accesible. Proporciona una interfaz clara y amigable para el usuario, lo que mejora la experiencia de uso y reduce significativamente los errores administrativos. Esta solución, al estar centrada en las necesidades reales de la empresa, demuestra cómo la programación puede resolver problemas concretos de forma efectiva.

Recomendaciones.

- Incorporar una base de datos: Sustituir los archivos TXT y JSON por una base de datos relacional (como SQLite o MySQL) permitiría una gestión más robusta, segura y escalable de la información.
- Implementar una interfaz gráfica de escritorio que permita usar botones, menús, ventanas, haciendo un sistema amigable e intuitivo.
- Habilitar múltiples usuarios con roles: Diseñar un sistema de acceso con diferentes niveles de permisos (por ejemplo, administrador, vendedor) permitiría un mejor control y trazabilidad de las acciones realizadas en el sistema.
- Planificar futuras integraciones: Pensar en una futura integración con servicios en la nube o sistemas de facturación electrónica podría ampliar las capacidades del sistema y prepararlo para un entorno más competitivo y profesional.
- Una mejora clave para la experiencia del cliente sería la incorporación de un sistema de envío automático de facturas mediante enlaces, dirigidos al número de teléfono que el cliente proporcione al momento de la venta. Esta solución digital no solo mejoraría la comodidad del usuario, sino que también reduciría el uso de papel y facilitaría la consulta de facturas en cualquier momento. Para ello, se podría vincular el sistema con servicios de mensajería como WhatsApp o SMS mediante APIs de terceros.

Referencias bibliográficas.

1. Bucles - Learn Python - free interactive Python tutorial. (s. f.). https://www.learnpython.org/es/Loops
2. Bustamante, S. J. (2021). *Sentencias If, Elif y Else en Python*. freeCodeCamp.org. https://www.freecodecamp.org/espanol/news/sentencias-if-elif-y-else-en-python/
3. Cimas, G. (2022). Visual Studio Code: Editor de código para desarrolladores. Open Webinars. https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/
4. Coursera. (2023). ¿Qué es Python y para qué se usa? Guía para principiantes. Coursera. https://www.coursera.org/mx/articles/what-is-python-used-for-a-beginners-guide-to-using-python
5. Gomila, J. G. (2024, 20 octubre). Set en Python, Elimina Automáticamente tus Archivos Duplicados. Frogames. https://cursos.frogamesformacion.com/pages/blog/set-en-python#:~:text=Un%20set%20en%20Python%20es,que%20elimina%20la%20redundancia%20autom%C3%A1ticamente
6. IPCISCO. (2023). Python String Methods | lower | upper | format | strip | join method IpCisco. IPCisco. https://ipcisco.com/lesson/python-string-methods/
7. Joyanes, L. (2020). Capítulo 6: Subprogramas (subalgoritmos): funciones) (pp. 209-237). En Fundamentos de Programación: Algoritmos, estructura de datos y objetos (5ta. Ed.). McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A. U.

8. Melov, F. (2022). *Sentencias Try y Except de Python: Cómo manejar excepciones en Python*. freeCodeCamp.org. https://www.freecodecamp.org/espanol/news/sentencias-try-y-except-de-python-como-manejar-excepciones-en-python/

Anexos.

Link al repositorio donde se encuentra alojado el proyecto por integrante:

Alexa Loaisiga: https://github.com/asloaisiga/sistemablacen

Chelsea Quintanilla: https://github.com/Chelsea270/Proyecto-final

Fabiola Lanuza: https://github.com/FBI1820/sistema-BLACEN

Luis Aguirre: https://github.com/Papaleta-ez/iz/tree/a7d3e3c136f5bb083f2db1d093a257d6052ca127/Trabajo%20Final

Link a la encuesta realizada: https://docs.google.com/forms/d/1oDv8SW4YPEqW4fial8wW2a_u0hNTt733yycvAZm_Wek/viewform

Link al google slides: SISTEMA BLACEN