**SISTEMA DE FACTURACIÓN ELECTRÓNICA (Noviembre de 2024)**

Autores: Torres Pachón Juan Esteban, González Gonzáles Dinalut y Ruiz Rincon Jessica Jineth estudiantes de la foundation universitaria de San gil unisangil (SEDE-CHIQUINQUIRA)

***Resumen*****—En el presente documento hemos diseñado un algoritmo un sistema de facturación electrónica.Por lo que aplicamos una metodología en la que establecimos los requisitos y limitaciones del algoritmo. Además utilizamos PSeInt para crear un pseudocódigo que luego implementamos en Python. Al ejecutar el programa logramos identificar que el algoritmo calcula de forma precisa el volumen de una pelota.**

**Palabras clave- Línea blanca,Pequeños electrodomésticos,Entretenimiento,Aires Acondicionados,Matrices,Funciones, Clases.**

INTRODUCCIÓN

 En este documento se busca diseñar un algoritmo donde se desarrolló un sistema básico de facturación electrónica donde se le permita al usuario gestionar un catálogo de productos, gestión venta de productos donde se imprime una factura

*ANÁLISIS*

1. *Contexto*

Una empresa necesita implementar un sistema de facturación electrónica para registrar ventas, gestionar el inventario y cumplir con las normativas fiscales.

*B. Población*

ElectroMundo S.A y clientes

*C. Limitaciones y alcance*

Las limitaciones y alcances que se pudieron identificar son las siguientes:

Limitaciones

* El usuario tiene que ingresar datos específicos.
* Tienen opciones limitadas tales con gestionar venta de productos tales como gestionar catálogo,hacer compras,hacer ventas, y agregar productos

Alcances

* El usuario podrá ingresar los datos requeridos.
* El usuario puede visualizar un catálogo.
* El usuario puede agregar productos.
* El usuario puede imprimir facturas.

III. OBJETIVOS

*Objetivo general*

Diseñar un algoritmo donde el usuario pueda gestionar facturas electrónica, registrar ventas, gestionar el inventario y cumplir con las normativas fiscales

1. *Objetivos específicos*

* Determinar el lenguaje de programación que se va a usar para crear el algoritmo.
* Solicitar al usuario los productos a vender o productos a agregar al inventario.
* imprimir una factura electrónica
* Generar la salida correspondiente de los datos.
* Presentar al usuario un programa que pueda manejar de forma comprensible.

IV. ESPECIFICACIÓN DE REQUERIMIENTOS.

1. *CON RESPECTO AL DISPOSITIVO*

A continuación se indican los requerimientos mínimos:

* Mínimo 4 GB de RAM.
* Sistema operativo: Windows 8 en adelante.
* CPU: Intel Core i3.
* Disco: HDD o SSD.

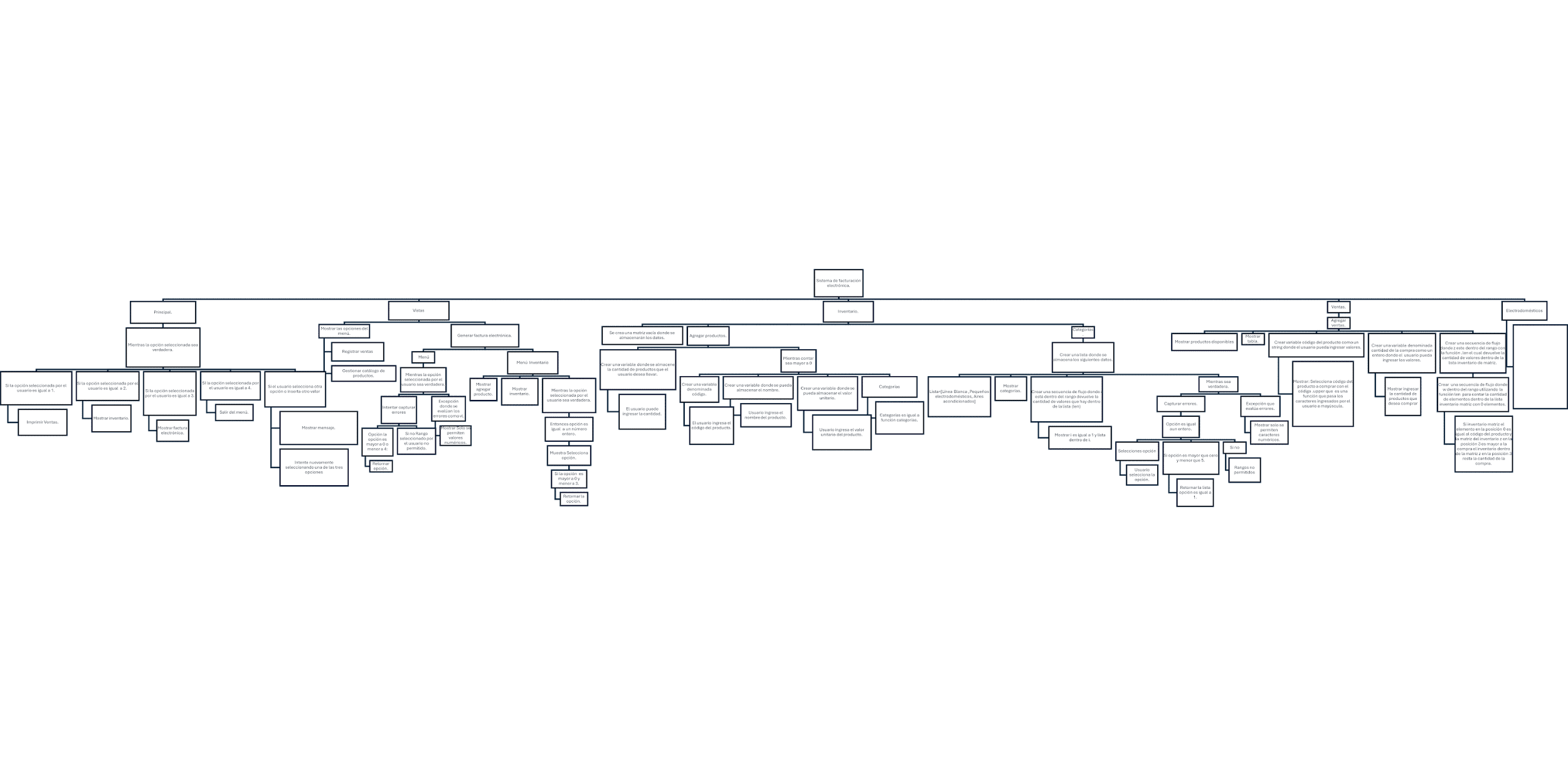
1. *CON RESPECTO AL USUARIO.*

*Requisitos mínimos*

* *El usuario debe ingresar los datos correspondientes para gestionar el inventario y registrar las ventas.*
* *El usuario debe contar con una necesidad seleccionada, la cual es la gestión del inventario y el registro de ventas*

V. DISEÑO DEL ALGORITMO

Utilizamos un programa generador de pseudocódigo denominado PSeInt, el cual utilizamos como guía, para luego usar el lenguaje de programación Python, siendo este el estructurador del software requerido y diseñamos mapa conceptual donde se muestra un análisis en base a la programacion modular.



*Ilustración 1.Análisis basado en la programación modular basado en el algoritmo.*

Tabla

Descripción generada automáticamente

*Ilustración 2. Clase dentro de la factura electrónica la cual se divide en atributos y método..*

VI.CODIFICACIÓN.

 Para la codificación del algoritmo utilizamos los siguientes  programas:

* PSeInt.
* Visual Studio Code.
* python
* Git hub.

*Instrucciones del código fuente*

1. *Código fuente*

Dentro del código fuente trabajamos con scripts donde cada script se conectaba con una función a continuación se anexan imágenes del código seccionado por módulos que complementan el script

**Main es donde se ejecuta el código:**

from View import \*

from Inventory import \*

from Sells import \*

from ElectroniF import \*

from ElectroniF import menu\_facturacion

print("ELECTRO MUNDO SAS")

while True:

option = menu()

#VENTAS

if option == 1:

while True:

menuVt = menuVentas()

if menuVt == 1:

agregarVenta()

elif menuVt == 2:

mostrarCantidadVendida()

elif menuVt == 3:

pass

confirm = str(input("Deseas continuar? S /N: "))

if confirm.upper() =="N":

break

#INVENTARIO

elif option == 2:

while True:

menuIn = menuInventory()

if menuIn == 1:

addProduct()

elif menuIn == 2:

showTable()

elif menuIn == 3:

break

confirm = str(input("Deseas continuar? S / N: "))

if confirm.upper() == "N":

break

elif option == 3:

menu\_facturacion()

pass

#Salir

**En el inventario almacenamos los datos del producto y funciones**

from tabulate import tabulate

import os

inventoryMatrix = []

def addProduct():

"""Función encargada de añadir un nuevo producto a la matriz del inventario."""

os.system('cls')

count = int(input("Cuantos productos deseeas ingresar?: "))

while count > 0:

os.system('cls')

codigo = str(input("Ingresa Código unico del producto: ")).upper()

print()

os.system('cls')

nombre = str(input("Ingresa nombre de producto: ")).upper()

print()

os.system('cls')

precio\_unitario = int(input("Ingresa precio unitario: "))

print()

os.system('cls')

categoria = categories()

print()

os.system('cls')

cantidadP = int(input("Ingrese Cantidad de producto: "))

print()

inventoryMatrix.append([codigo, nombre, precio\_unitario, cantidadP ,categoria])

count -= 1

def showTable():

"""Función encargada de mostrar el inventario"""

headers = ["Código", "Nombre", "Precio Unitario", "Stock", "Categoria"]

print(tabulate(inventoryMatrix, headers=headers, tablefmt="fancy\_grid"))

def categories():

"""Función encargada de mostrar las categorias y retornar la selección."""

listC = ["Línea Blanca", "Pequeños Electrodomésticos", "Entretenimiento", "Aires Acondicionados"]

print("Categorias:\n")

for z in range(len(listC)):

print(f"{z + 1}. {listC[z]}\n")

while True:

try:

option = int(input("Selecciona opción: "))

if option > 0 and option < 5:

return listC[option - 1]

else:

print("\nRangos no permitidos!\n")

except ValueError as vl:

print("\nSolo se permiten caracteres numúricos!\n")

**Venta se encarga de generar la lógica con respecto al comercio de los productos.**

from Inventory import \*

from tabulate import tabulate

import os

matrixCantidadVendida = []

def agregarVenta():

"""Función encargada de realizar la compra, y interactuar con la matriz del archivo: Inventory.py."""

os.system('cls')

print("\nProductos disponibles!\n")

showTable()

codeProduct = str(input("Selecciona código de producto a comprar: ")).upper()

cantidadCompra = int(input("Ingresa la cantidad que deseas comprar: "))

for z in range(len(inventoryMatrix)):

if inventoryMatrix[z][0] == codeProduct and inventoryMatrix[z][3] > cantidadCompra:

inventoryMatrix[z][3] -= cantidadCompra

cantidadVent(codeProduct, cantidadCompra)

def cantidadVent(code, cantidad):

"""Función encargada de registrar las ventas de cada producto."""

if not matrixCantidadVendida:

matrixCantidadVendida.append([code, cantidad])

else:

for z in range(len(matrixCantidadVendida)):

if matrixCantidadVendida[z][0] != code:

matrixCantidadVendida.append([code, cantidad])

else:

matrixCantidadVendida[z][1] += cantidad

def mostrarCantidadVendida():

"""Función encargada de mostrar la cantidad vendida de productos."""

header = ["Código", "Cantidad Vendida"]

print(tabulate(matrixCantidadVendida, headers=header, tablefmt="fancy\_grid"))

**Electronic se encarga de generar la factura**

from datetime import datetime

from tabulate import tabulate

# Lista para almacenar las facturas en memoria

facturas = []

class GenerarFactura:

def \_\_init\_\_(self, nombre, cantidad, precio):

self.nombre = nombre

self.cantidad = cantidad

self.precio = precio

self.subtotal = self.cantidad \* self.precio

def mostrar\_producto(self):

return [self.nombre, self.cantidad, self.precio, self.subtotal]

def mostrar\_detalles(productos):

print("\nDetalles de los productos comprados:")

headers = ["Nombre", "Cantidad", "Precio", "Subtotal"]

table = [producto.mostrar\_producto() for producto in productos]

print(tabulate(table, headers=headers, tablefmt="fancy\_grid"))

def calcular\_totales(productos):

subtotal = sum(producto.subtotal for producto in productos)

iva = subtotal \* 0.19 # IVA del 19%

total = subtotal + iva

return subtotal, iva, total

def guardar\_factura(productos, subtotal, iva, total):

factura = {

"productos": [producto.\_\_dict\_\_ for producto in productos],

"subtotal": subtotal,

"iva": iva,

"total": total,

"fecha": datetime.now().strftime("%d/%m/%Y %H:%M:%S")

}

facturas.append(factura)

print("\nFactura guardada para futuras consultas.")

def mostrar\_facturas():

if not facturas:

print("No hay facturas guardadas.")

return

for i, factura in enumerate(facturas):

print(f"\nFactura {i + 1}:")

print(f"Fecha: {factura['fecha']}")

print("\nDetalles de los productos comprados:")

headers = ["Nombre", "Cantidad", "Precio", "Subtotal"]

table = [[p['nombre'], p['cantidad'], p['precio'], p['subtotal']] for p in factura["productos"]]

print(tabulate(table, headers=headers, tablefmt="fancy\_grid"))

print(f"\nSubtotal: {factura['subtotal']:.2f}")

print(f"IVA: {factura['iva']:.2f}")

print(f"Total: {factura['total']:.2f}")

def menu\_facturacion():

productos = []

while True:

print("\nMenú de Facturación:")

print("1. Ingresar producto")

print("2. Mostrar detalles de la factura")

print("3. Calcular y mostrar totales")

print("4. Guardar factura")

print("5. Mostrar facturas guardadas")

print("6. Volver al menú principal")

opcion = input("Seleccione una opción (1-6): ")

if opcion == "1":

nombre = input("Ingrese el nombre del producto: ")

cantidad = int(input("Ingrese la cantidad: "))

precio = float(input("Ingrese el precio: "))

producto = GenerarFactura(nombre, cantidad, precio)

productos.append(producto)

elif opcion == "2":

if productos:

mostrar\_detalles(productos)

else:

print("No hay productos ingresados.")

elif opcion == "3":

if productos:

subtotal, iva, total = calcular\_totales(productos) print(f"\nSubtotal:{subtotal:.2f}, IVA: {iva:.2f}, Total: {total:.2f}")

else:

print("No hay productos ingresados.")

elif opcion == "4":

if productos:

subtotal, iva, total = calcular\_totales(productos)

guardar\_factura(productos, subtotal, iva, total)

productos.clear() # Limpiar productos después de guardar la factura

else:

print("No hay productos ingresados.")

elif opcion == "5":

mostrar\_facturas()

elif opcion == "6":

break

else:

print("Opción inválida, por favor ingrese una opción válida.")

**vista es el encargado de mostrar el menú de opciones**

**import os**

**def menu():**

**"""Función encargada de presentar el menú de opciones."""**

**os.system('cls')**

**print("\n1.) Registrar venta\n\n2.) Gestionar Catalogo de productos\n\n3.) Generar Factura Electronica.\n")**

**while True:**

**try:**

**option = int(input("Selecciona opción: "))**

**print()**

**if option > 0 and option < 4:**

**return option**

**else:**

**print("\nRango no permitido!\n")**

**except ValueError as vl:**

**print("\nSolo se Permiten Caractfdseres numéricos!\n")**

**def menuVentas():**

**"""Función encargada de retornar la selección**

**del menú de ventas"""**

**os.system('cls')**

**print("1.) Comprar\n\n2.) Cantidades vendidas\n\n3.) Salir\n")**

**while True:**

**try:**

**option = int(input("Selecciona opción: "))**

**if option > 0 and option < 4:**

**return option**

**else:**

**print("\nRango no permitido!\n")**

**except ValueError as vl:**

**print("\nSolo se Permiten Caracteres numéricos!\n")**

**def menuInventory():**

**"""Función encargada de retornar la selección del menú**

**de inventario."""**

**os.system('cls')**

**print("1.) Agregar Producto\n\n2.) Mostrar Inventario\n\n3.) Salir\n")**

**while True:**

**try:**

**option = int(input("Selecciona opción: "))**

**if option > 0 and option < 4:**

**return option**

**else:**

**print("\nRango no permitido!\n")**

**except ValueError as vl:**

**print("\nSolo se Permiten Caracteres numéricos!\n")**

Imagen. 3. Código fuente.

*Imagen. 2:En esta imágenes se muestra el diseño del algoritmo donde se utiliza el lenguaje de programación python para ejecutarlo .*

*VII. EJECUCIÓN DEL PROGRAMA.*

En este caso comenzamos con la apertura de la terminal de Git Bash. Si ya se creó el repositorio remoto y se sincronizan las cuentas de los desarrolladores con el repositorio local, se envía el archivo al Staging área con el comando git add.  Luego ejecutamos el git commit -m “Comentario” para confirmar los cambios, donde, posteriormente usamos git push origin o pb master o main dependiendo la rama que se esté usando con el fin de almacenar la información en el repositorio remoto de GitHub.

VIII. VERIFICACIÓN Y DEPURACIÓN

En el versionamiento de la aplicación iniciando desde la versión -V 0.1. Este código inicial fue la base estructural con la cual mediante un análisis anteriormente realizado, se pudo establecer los requisitos que debe cumplir el programa para funcionar según lo dictaminado por el usuario final, siendo este la empresa ElectroMundo S.A.S y Clientes.

Ahora el código se encuentra en la versión -V 0.5 teniendo 5 archivos principales, los cuales desde ellos se maneja tanto la lógica e interfaz gráfica de la aplicación.

El primer archivo que se maneja es Inventory.py: Consta de 3 funciones:

addProduct ( ): Función encargada de añadir un nuevo producto a la matriz del inventario.

showTable ( ): Función encargada de mostrar el inventario.

categories ( ): Función encargada de mostrar las categorías y retornar la selección.

El segundo archivo manejado es Sells.py este archivo consta de 3 funciones:

agregarVenta ( ): Función encargada de realizar la compra e interactuar con la matriz del archivo Inventory.py.

cantidadVent ( ): Función encargada de registrar las ventas de cada producto.

mostrarCantidadVendida ( ): Función encargada de mostrar la cantidad vendida de los productos.

El tercer archivo manejado es View.py este archivo consta de 3 funciones:

menu ( ): Función encargada de presentar el menú principal de opciones

menuVentas ( ): Función encargada de retornar la selección del menú de ventas.

menuInventory ( ): Función encargada de retornar la selección del menú del inventario.

El cuarto archivo manejado se llama Electronic.py, este archivo consta de una clase, y cinco funciones.

1. ***Interna.***

*Toda la información relacionada con el desarrollo del código se encuentra almacenada en un repositorio en GitHub.*

*Enlace:* [*https://github.com/Zheilon/Parcial\_corte\_segundo.git*](https://github.com/Zheilon/Parcial_corte_segundo.git)

***Descripción del problema:***

Se identificó l necesidad de diseñar un algoritmo Desarrollar un sistema básico de facturación electrónica que permita:

***Modelos de análisis:***

Como modelo de análisis podemos identificar que se utilizó la programación modular para analizar el problema

Manual de usuario.

El usuario principalmente visualizará un menú dentro del cual puede seleccionar varias opciones.

1.) Registrar venta

2.) Gestionar Catálogo de productos

3.) Generar Factura Electrónica.

Selecciona opción:

Si selecciona la opción uno, se le despliegan otras opciones:

1.) Comprar

2.) Cantidades vendidas

3.) Salir

Donde puede seleccionar algunas de las opciones.

Si selecciona la opción 2 Gestionar el catálogo de productos.

1.) Agregar Producto

2.) Mostrar Inventario

3.) Salir

Selecciona opción:

y si selecciona la opcion 3.

Menú de Facturación:

1. Ingresar producto

2. Mostrar detalles de la factura

3. Calcular y mostrar totales

4. Guardar factura

5. Mostrar facturas guardadas

6. Volver al menú principal

Seleccione una opción (1-6):

VIII CONCLUSIONES.

* En el presente trabajo logramos describir el  diseño y codificación de un algoritmo el cual permite al usuario calcular el volumen de una pelota inflable para saber cuánto material usar para la fabricación del producto.
* Se desarrolló un fase de planeación la cual nos permitió garantizar una buena gestión de las fases del proyecto, tomando en cuenta fases como: requerimientos del cliente, el análisis y diseño de la estructura de la aplicación, la respectiva codificación teniendo en cuenta la fase anterior y por último la respectiva documentación aquí escrita.
* Implementamos un análisis situacional específico para captar con mayor detalle los requisitos del cliente,el pudimos utilizar para escribir el código de manera correcta
* Incorporamos la evaluación y pruebas para el programa, con ellas constatamos y verificamos que el código realizado por nosotros los desarrolladores esté a fin con los requerimientos del cliente.
* Aprendimos que es necesario aplicar una metodología que hace más comprensible y efectivo el proceso a la hora de desarrollar un algoritmo.
* Empleamos un generador de pseudocódigo denominado PSeInt como guía para estructurar de una forma clara el algoritmo en Python.
* Para realizar la codificación del código utilizamos varios programas como Visual Studio Code y GitHub los cuales facilitan el trabajo en grupo y la gestión de información.