SIGESA

ACME EDUCATION

Daniel Rodriguez.
2024

Campuslands Bucaramanga

SISTEMA DE GESTIÓN DE ASISTENCIA ACADÉMICA

Contexto General

ACME Education, una institución educativa dedicada a la formación técnica y profesional ha decidido modernizar su sistema de gestión de asistencia. Este sistema permitirá un control automatizado de la asistencia de sus estudiantes, facilitando la generación de informes que apoyen la mejora continua tanto de los procesos académicos como administrativos.

Para este fin, ACME Education ha encargado el desarrollo de una solución informática llamada **Sistema de Gestión de Asistencia Académica (SISGESA)**. Este sistema se implementará como un programa de consola, permitiendo a los usuarios interactuar con las diversas funciones a través de un menú.

Descripción del Problema

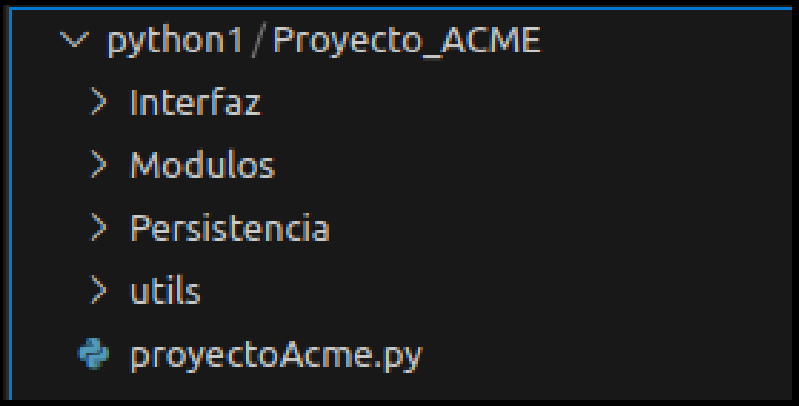
El registro manual de la asistencia de los estudiantes ha generado ineficiencias en la obtención de información fiable, afectando tanto los procesos académicos como administrativos. La solución propuesta debe automatizar este proceso, permitiendo el acceso a la información relevante y generando informes de manera clara y precisa.

Objetivos del Proyecto

Se solicita el desarrollo de un programa para la **consola o terminal** y escrito en el lenguaje de programación **Python**, que funcione como una aplicación de consola, para gestionar la asistencia académica. El sistema debe cumplir con los siguientes requisitos:

ESTRATEGIAS

La primera estrategia planteada para la resolución del problema fue: la fragmentación del programa, con el fin de llevar de manera ordenada todos los datos que me ofrece el ejercicio.



El fragmentar los datos dan el primer vistazo de como debe de ir ordenado el documento.
Con esto podrá seguirse un rumbo y desarrollarse por partes.

El primer punto planteado en el ejercicio, se nos habla de que habrá una entrada por parte del usuario, el cual sera el ingreso al aplicativo, donde tendrá una contraseña predeterminada que podrá cambiarse en cualquier momento y sera encriptada.

1. **Inicio de Sesión**:
   - Al iniciar el programa, debe solicitarse un nombre de usuario y contraseña. La primera vez que el sistema sea ejecutado, la contraseña será la predefinida: "SISGESA".
   - El sistema debe permitir cambiar esta contraseña a través de una opción en el menú.
   - La contraseña debe ser guardada en un archivo y asegurada mediante **algoritmos de encriptación nativos de Python**, como **SHA-256**, sin la necesidad de instalar módulos adicionales.

USO DE FUNCIONES

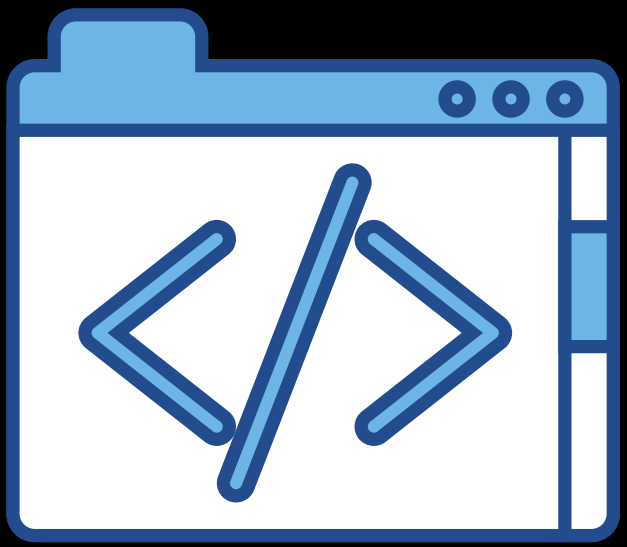
La forma mas adecuada para desarrollar ejercicios que pueden llegar a ser complejos y largos, es el uso de funciones y la importación de módulos.

A continuación se puede observar la declaración de la función de ContrasenaIni() como la contraseña predeterminada que regirá la entrada de los usuarios hasta que por elección del usuario la cambie, en ella se importa a un archivo que protege la información después de cerrar el programa.

```
def contrasenaIni():
    try:
        with open("Proyecto_ACME/utils/Contraseña.json", "r") as fd:
            return json.load(fd)
    except FileNotFoundError:
        return "SISGESA"
```

```python
def verificar_contrasena(ingresada):
    return ingresada == contrasenaIni()
```

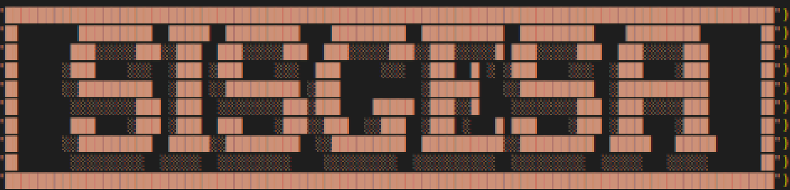
Se declarara una funcion que se encargue de comparar la contraseña ingresada por parte del usuario con la que se tiene en el almacenamiento



```python
def login():
    while True:

        print(Fore.LIGHTRED_EX+"
        print(Fore.LIGHTRED_EX+"~
        print(Fore.LIGHTRED_EX+"~
        print(Fore.LIGHTRED_EX+"~
        print(Fore.LIGHTRED_EX+"~
        print(Fore.LIGHTRED_EX+"~
        print(Fore.LIGHTRED_EX+"~
        print(Fore.LIGHTRED_EX+"~
        print(Fore.LIGHTRED_EX+"~
        print(Fore.LIGHTRED_EX+"~
        print(Fore.LIGHTRED_EX+"~
        print(Fore.LIGHTRED_EX+"~
        print(Fore.LIGHTRED_EX+"~
        print(Fore.LIGHTRED_EX+"~
        print(Fore.LIGHTRED_EX+"~
        print(Fore.LIGHTRED_EX+"~
        print(Fore.LIGHTRED_EX+"~
        print(Fore.LIGHTRED_EX+"
```

```python
        input("Precione Enter tecla para continuar...")
        limpiarPantalla
        print("BIENVENIDO")
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
        print(Fore.BLACK+"
```

```python
        input("Precione Enter para continuar...")
        limpiarPantalla

        try:

            print(Fore.LIGHTMAGENTA_EX+"
            print(Fore.LIGHTMAGENTA_EX+"
            print(Fore.LIGHTMAGENTA_EX+"
            print(Fore.LIGHTMAGENTA_EX+"
            print(Fore.LIGHTMAGENTA_EX+"
            print(Fore.LIGHTMAGENTA_EX+"
            print(Fore.LIGHTMAGENTA_EX+"
            print(Fore.LIGHTMAGENTA_EX+"
            print(Fore.LIGHTMAGENTA_EX+"
            print(Fore.LIGHTMAGENTA_EX+"
```



```python
            print("Si es tu primer ingreso la clave es SISGESA")



            print("="*40)
            usuario = input(">>Ingrese el nombre de usuario: ")
            contrasena = input(">>Ingresa la contraseña: ")
            print("="*40)
            if verificar_contrasena(contrasena):
                print("La contraseña indicada es correcta.")
                break
            else:
                print(Fore.RED+">>Contraseña incorrecta.")
        except Exception as e:
            print(Fore.RED+">>Error. Contraseña no valida")
```

declaracion de la funcion login(), la que se encarga de dar ingreso y abrir el programa a el usuario, en este caso se hizo un poco mas interactivo aprovechando diferentes herramientas

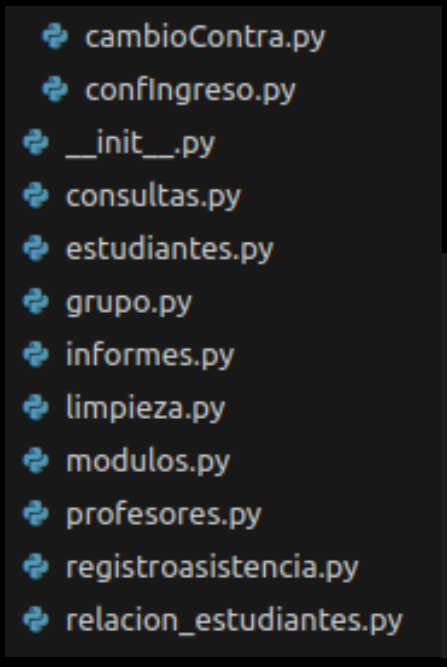
Este es el cerebro de la funcion. la encargada de usar las funciones anteriores y emplearlas para generar el ingreso.

```
    print("Si es tu primer ingreso la clave es SISGESA")



    print("="*40)
    usuario = input(">>Ingrese el nombre de usuario: ")
    contrasena = input(">>Ingresa la contraseña: ")
    print("="*40)
    if verificar_contrasena(contrasena):
        print("La contraseña indicada es correcta.")
        break
    else:
        print(Fore.RED+">>Contraseña incorrecta.")
except Exception as e:
    print(Fore.RED+">>Error. Contraseña no valida")
```

En la siguiente parte del ejercicio se encuentra un menú que sera usado con diferentes fines.

2. **Menú de Opciones**:
   - El programa debe presentar un menú con opciones claras para interactuar con el sistema. Las opciones del menú deben incluir:
     a. Registro de grupos.
     b. Registro de módulos.
     c. Registro de estudiantes.
     d. Registro de docentes.
     e. Registro de asistencia.
     f. Consultas de información.
     g. Generación de informes.
     h. Cambio de contraseña.
     i. Salida del sistema.

Es importante ser ordenado con los módulos que se exportaran mas adelante, para dejar el código base lo mas pulcro posible.



El documento __init___.py es un documento que se creo con el fin de solucionar un problema que perduro por horas

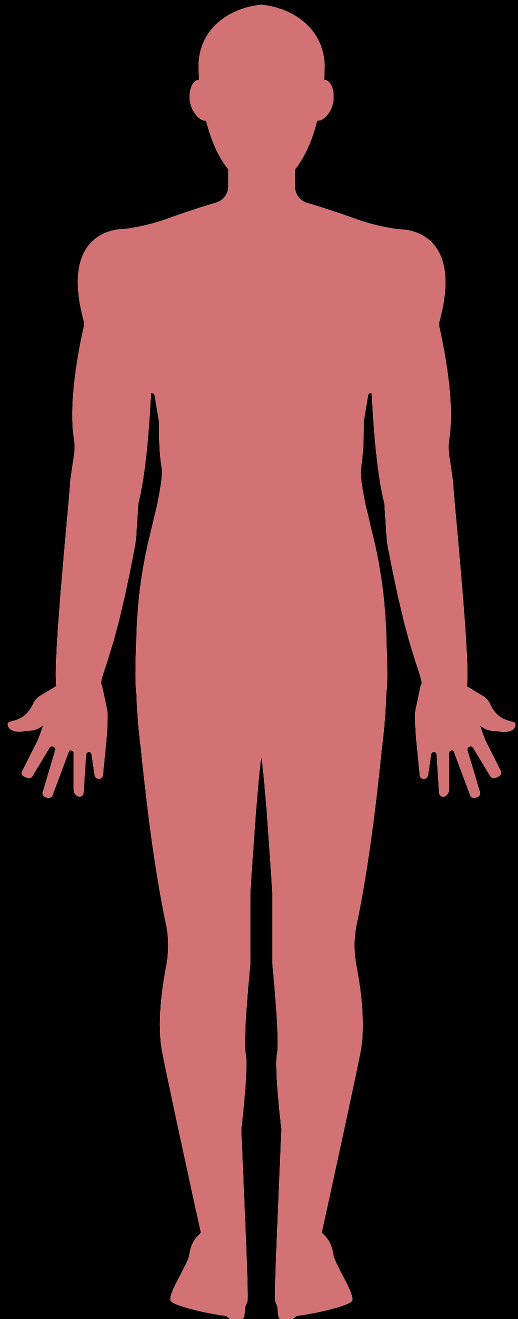
```
from Modulos.limpieza import limpiarPantalla
from Modulos.estudiantes import registroEstudiante
from Interfaz.menu import menu
from Modulos.grupo import registroGrupo
from Modulos.modulos import  registroModulo
from Modulos.profesores import registroDocente
from Modulos.registroasistencia import registroAsistencia
from Modulos.ingreso_contra.confIngreso import login
from Modulos.ingreso_contra.cambioContra import cambiarContrasena
from Modulos.relacion_estudiantes import relacionarEstudiantes
from Persistencia.persistencia import guardar
from Persistencia.cargar import cargar_archivo_json
from Modulos.consultas import consultarDocentesPorModulo, consultarEstudiantesPorDocente
from colorama import Fore, Style, init
```

Las estructura del menú fue divida en dos, una parte visual que el usuario vera y la parte que se encargara de gestionar cada una de sus funciones, puede compararse al ser humano, el cuerpo que todos vemos y cerebro que lo piensa.

```python
login()
limpiarPantalla()
while True:
    opc = menu()
    match opc:
        case "a":
            limpiarPantalla()
            guardar(registroGrupo(cargar_archivo_json("grupos")),"grupos")

        case "b":
            limpiarPantalla()
            guardar(registroModulo(cargar_archivo_json("modulos")), "modulos")

        case "c":
            limpiarPantalla()
            guardar(registroEstudiante(cargar_archivo_json("estudiantes")),"estudiantes"),

        case "d":
            guardar(relacionarEstudiantes(cargar_archivo_json("estudiantes"),cargar_archivo_json("modulo"),cargar_archivo_json("grupos"),cargar_archivo_json("asignacion")),
            "asignacion")
        case "e":
            limpiarPantalla()
            guardar(registroDocente(cargar_archivo_json("docentes"), cargar_archivo_json("modulos")), "docentes")

        case "f":
            limpiarPantalla()
            guardar(registroAsistencia(cargar_archivo_json("estudiantes"),cargar_archivo_json("modulo"),cargar_archivo_json("asistencia")),"asistencia")

        case "g":
            limpiarPantalla()
            # consultasPorCodigo(cargar_archivo_json("asignacion"),cargar_archivo_json("Estudiantes"))
            opc=menuConsulta()
            match opc:
                case"a":
                    guardar(consultarEstudiantesPorGrupo(cargar_archivo_json("estudiantes"),cargar_archivo_json("grupo")),"consultaEstudiantePorGrupo"),
                case"b":
                    guardar(consultarEstudiantesPorModulo(cargar_archivo_json("estudiantes"),cargar_archivo_json("modulo")),"consultaEstudiantePorModulo"),
                case"c":
                    guardar(consultarDocentesPorModulo(cargar_archivo_json("modulo"),cargar_archivo_json("docente")),"consultaDocentePorModulo"),
                case"d":
                    guardar(consultarEstudiantesPorDocente(cargar_archivo_json("profesor"),cargar_archivo_json("modulo"),cargar_archivo_json("estudiantes")),"ConsultaEstudiantePorDocente")
        case "h":
            pass

        case "i":
            cambiarContrasena()

        case "j":
            print("Gracias por usar el software")


            print(Fore.GREEN + "*[illegible]*")
            print(Fore.GREEN + "88[illegible]88")
            print(Fore.GREEN + "88[illegible]88")
            print(Fore.GREEN + "88[illegible]88")
            print(Fore.GREEN + "88[illegible]88")
            print(Fore.GREEN + "88[illegible]88")
            print(Fore.GREEN + "88[illegible]88")
            print(Fore.GREEN + "88[illegible]88")
            print(Fore.GREEN + "88[illegible]88")
            print(Fore.GREEN + "88[illegible]88")
            print(Fore.GREEN + "88[illegible]88")
            print(Fore.GREEN + "88[illegible]88")
            print(Fore.GREEN + "88[illegible]88")
            print(Fore.GREEN + "88[illegible]88")
            print(Fore.GREEN + "*[illegible]*")



            break
```

Este es el cuerpo visual del programa

```python
from colorama import Fore, Style, init
from Modulos.limpieza import limpiarPantalla
def menu():
    limpiarPantalla()
    while True:

        init(autoreset=True)



        print(Fore.RED +"[illegible]")
        print(Fore.RED +"[illegible]")
        print(Fore.RED +"[illegible]")
        print(Fore.RED +"[illegible]")
        print(Fore.RED +"[illegible]")
        print(Fore.RED +"[illegible]")
        print(Fore.RED +"[illegible]")
        print(Fore.RED +"[illegible]")
        print(Fore.RED +"[illegible]")
        print(Fore.RED +"[illegible]")
        print(Fore.RED +"[illegible]")
        print(Fore.RED +"[illegible]")
        print(Fore.RED +"[illegible]")
        print(Fore.RED +"[illegible]")
        print(Fore.RED +"[illegible]")
        print(Fore.RED +"[illegible]")

        print(""*10)
        print("="*40)
        print("a. Registro de grupos.")
        print("b. Registro de módulos.")
        print("c. Registro de estudiantes.")
        print("d. Relacionar.")
        print("e. Registro de docentes.")
        print("f. Registro de asistencia.")
        print("g. Consultas de información.")
        print("h. Informe asistencia.")
        print("i. Cambio de contraseña.")
        print("j. Salida del sistema.")
        print("="*40)
        print("")
        print(">> Ingresar a opcion: ", end="")
        try:
            opciones = ["a", "b", "c","d","e","f", "g", "h", "i"]
            opc = input().lower()
            if opc not in opciones:
                print(Fore.RED+"Error opcion invalida")
                print("presione cualquier tecla para volver al menu. \n")
            return opc

        except ValueError:
            print(Fore.RED+"ingreso una opcion incorrecta")
            print("presione cualquier tecla para volver al menu. \n")
```



Se tomo consideración con respecto a la revisión de errores, con el fin de poder hacer un codigo lo mas robusto posible.

En este caso, cada función de este modulo esta protegida contra ciertos errores.

```python
def leerCodigoModulo():
    while True:
        try:
            cod = int(input("Ingrese el código del módulo: "))
            if cod:
                return cod

            print(Fore.RED+">>> Error. Código inválido")
        except ValueError:
            print(Fore.RED+">>Error. Codigo invalido")

def nombreModulo():
    while True:
        try:
            strNombreModulo = input("Ingrese nombre del módulo: ").strip()
            if strNombreModulo:
                return strNombreModulo
            print(Fore.RED+">>> Error. Nombre inválido")
        except Exception as e:
            print(Fore.RED+">>Error al iniciar el nombre. ")

def duracionModulo():
    while True:
        try:
            dModulo1 = int(input("Ingrese la duración del módulo (semanas): "))
            if dModulo1 > 0:
                return dModulo1
            print(Fore.RED+">>> Error. Duración inválida.")
        except ValueError:
            print(Fore.RED+">>> Error. Duración inválida.")

def horaInicio():
    while True:
        try:
            hora = input("Ingrese la hora de inicio de la clase (HH:MM): ")
            datetime.strptime(hora, "%H:%M")  # Validar formato de hora
            return hora
        except ValueError:
            print(Fore.RED+">>> Error. Formato de hora inválido. Debe ser HH:MM.")
```

La persistencia de datos

Este parte del código, es una de las mas importantes para el ejercicio que se intentaba solucionar, ya que al ser un programa que almacenara datos a largo plazo, se necesitaba de una función que encargara de guarda la información.

```python
import json


def guardar(datos, nombre):
    ruta_archivo = f'python1/Proyecto_ACME/utils/{nombre}.json'


    try:
        with open(ruta_archivo, 'r') as fd:
            contenido = json.load(fd)
            if not isinstance(contenido, dict):
                contenido = {}
    except FileNotFoundError:
        contenido = {}


    if datos:
        contenido.update(datos)


    with open(ruta_archivo, 'w') as fd:
        json.dump(contenido, fd, indent=4)
```

CARGAR Y REVISAR ARCHIVOS

Cargar un archivo también hace parte de la persistencia, esta función de tamaño pequeño, es esencial, a lo largo del ejercicio, ya que estaremos revisando los datos guardados para poder seguir desarrollando otros puntos del ejercicio

```python
import json


def cargar_archivo_json(nombre_archivo):
    ruta = f'python1/Proyecto_ACME/utils/{nombre_archivo}.json'
    try:
        with open(ruta, 'r') as archivo:
            return json.load(archivo)
    except FileNotFoundError:
        print(f"")
        return {}
```

estos dos códigos siento que eran la base para la resolución de la mayoría de lo puntos que nos pedían en el menú.

```python
import json

def buscarGrupo(grupo):
    codGrupo = input("Ingrese el código del grupo: ")
    if codGrupo in grupo:
        return codGrupo
    else:
        print("Grupo no existe")
        return None

def asignarModulo(modulo):
    cont = int(input("¿Cuántos módulos desea ingresar? "))
    if cont > 3:
        print("maximo de modulos son 3")
    else:
        modulos_asignados = {}
        print(modulo)
        for i in range(1, cont + 1):
            codModulo = input(f"Ingrese el código del módulo {i}: ")
            if codModulo in modulo:
                modulos_asignados[f"m{i}"] = codModulo
            else:
                print("Módulo no existe")

    return json.dumps(modulos_asignados)

def buscarEstudiante(estudiantes):
    codEstudiante = input("Ingrese el código del estudiante: ")
    if codEstudiante in estudiantes:
        return codEstudiante
    else:
        print("Estudiante no existe")


def relacionarEstudiantes(estudiantes, modulos, grupos, datos):
    datosEstudiante = buscarEstudiante(estudiantes)
    print(datosEstudiante)
    if datosEstudiante is None:
        print("El estudiante no esta registrado")

    if datosEstudiante not in datos:
        datosGrupo = buscarGrupo(grupos)
        print(datosGrupo)
        if datosGrupo is None:
            print("El grupo no esta registrado")
        datosAsignacion = {
            "codigo": datosEstudiante,
            "Grupo": datosGrupo,
            "modulo": json.loads(asignarModulo(modulos))
        }
        datos[datosEstudiante] = datosAsignacion
        print(f"Asignación completada para el estudiante {datosEstudiante}.")
    else:
        print("Estudiante ya asignado.")

    return datos
```

```python
from colorama import Fore, Style, init
from datetime import datetime
init(autoreset=True)


def leerCodigoModulo():
    while True:
        try:
            cod = int(input("Ingrese el código del módulo: "))
            if cod:
                return cod

            print(Fore.RED+">>> Error. Código inválido")
        except ValueError:
            print(Fore.RED+">>Error. Codigo invalido")

def nombreModulo():
    while True:
        try:
            strNombreModulo = input("Ingrese nombre del módulo: ").strip()
            if strNombreModulo:
                return strNombreModulo
            print(Fore.RED+">>> Error. Nombre inválido")
        except Exception as e:
            print(Fore.RED+">>Error al iniciar el nombre. ")

def duracionModulo():
    while True:
        try:
            dModulo1 = int(input("Ingrese la duración del módulo (semanas): "))
            if dModulo1 > 0:
                return dModulo1
            print(Fore.RED+">>> Error. Duración inválida.")
        except ValueError:
            print(Fore.RED+">>> Error. Duración inválida.")

def horaInicio():
    while True:
        try:
            hora = input("Ingrese la hora de inicio de la clase (HH:MM): ")
            datetime.strptime(hora, "%H:%M")  # Validar formato de hora
            return hora
        except ValueError:
            print(Fore.RED+">>> Error. Formato de hora inválido. Debe ser HH:MM.")

def horaFin():
    while True:
        try:
            hora = input("Ingrese la hora de fin de la clase (HH:MM): ")
            datetime.strptime(hora, "%H:%M")  # Validar formato de hora
            return hora
        except ValueError:
            print(Fore.RED+">>> Error. Formato de hora inválido. Debe ser HH:MM.")

def registroModulo(modulo):
    codigo = leerCodigoModulo()
    if codigo not in modulo:
        nombre = nombreModulo()
        duracion = duracionModulo()
        inicio = horaInicio()
        fin = horaFin()
        modulo2 = {
            "Nombre": nombre,
            "Duracion": duracion,
            "Hora_Inicio": inicio,
            "Hora_Fin": fin
        }
        modulo[codigo]=modulo2
        print("Módulo registrado correctamente.")
    else:
        print("El código del módulo ya existe.")

    input("Presione cualquier tecla para volver al menú...")
    return modulo
```

¡MUCHAS GRACIAS POR SU ATENCIÓN!