

Sistema de Gestión de Asistencia Académica (SISGESA).

Por: Jennifer Arciniegas Arciniegas

A dark blue diagonal gradient bar that starts from the bottom left corner and extends towards the top right corner, covering the lower half of the slide.

Objetivos

Objetivo general:

Permitir un control automatizado de la asistencia que facilite la generación de informes que apoyen los procesos académicos como administrativos.

Objetivos específicos:

Realizar un programa para consola o terminal en lenguaje Python.

Permitir la validación de un usuario y contraseña

Permitir realizar ingreso y consultas

login

Valida que los datos ingresados por el usuario sean los mismo que se alojan en el documento json, incluso cuando se cambia de contraseña

```
SISGESA > archivo > {} cuentajson > ...
```

```
1  {
2    "administrados": {
3      "user": "admin",
4      "password": "2fc2c9f066f015d7b9b43ed16cb07ff19724ec3d3876ccb1f3e5276a0dd65568"
5    }
6  }
```

```
SISGESA > modelo > iniciosesion.py > login
```

```
1  import json
2  import hashlib
3
4  def consultaruser():
5      with open ("SISGESA/archivo/cuenta.json", "r") as archivo:
6          dato = json.load(archivo)
7          return dato.get("administrados")
8
9  > def consultarclave():--
13
14  > def leerPassword():--
24
25  > def leerUser():--
34
35
36  def login(): #funcion para cargar credenciales almacenadas
37      user = leerUser()
38      administrador = consultaruser()
39
40      if user == administrador.get('user'):
41          userpassword = leerPassword()
42          if userpassword == administrador.get('password'):
43              print("contraseña correcta")
44              return administrador
45          else:
46              print("la contraseña es incorrecta")
47              return None
48      else:
49          print("el usuario es incorrecto")
50          return None
51
52
53  def cambiarpassword():
54      user = leerUser()
55      with open ("SISGESA/archivo/cuenta.json", "r") as archivo:
56          dato = json.load(archivo)
57          newPass = input("ingrese la nueva contraseña:\n")
58          dato["administrados"]["password"] = hashlib.sha256(newPass.encode('utf-8')).hexdigest()
59          with open("SISGESA/archivo/cuenta.json", 'w') as archivo:
60              json.dump(dato, archivo, indent=4)
61          print("contraseña actualizada")
62
```

Index y manejo de módulos

```
PROJECT
└─ SISGESA
   └─ archivo
      { } asignacion.json
      { } asistencia.json
      { } cuenta.json
      { } Docentes.json
      { } Estudiantes.json
      { } grupos.json
      { } Modulo.json
   └─ interfaz
      └─ menu.py
   └─ modelo
      └─ asignacion.py
      └─ consultaPorCodigo.py
      └─ iniciosesion.py
      └─ R_Asiistencia.py
      └─ R_Docentes.py
      └─ R_Estudiantes.py
      └─ R_Grupo.py
      └─ R_Modulo.py
   └─ persistencia
      └─ cargar.py
      └─ pesistenciaGuardar.py
   └─ index.py M

SISGESA > index.py > ...
1  from modelo.consultaPorCodigo import consultasPorCodigo
2  from interfaz.menu import menu
3  from modelo.R_Grupo import registroGrupos
4  from modelo.R_Estudiantes import Re_estudiantes
5  from persistencia.pesistenciaGuardar import guardar
6  from modelo.R_Modulo import registroModulo
7  from persistencia.cargar import *
8  from modelo.R_Docentes import registroDocentes
9  from modelo.iniciosesion import login
10 from modelo.iniciosesion import cambiarpassword
11 from modelo.R_Asiistencia import registroAsistencia
12 from modelo.asignacion import asignacion
13
14 if login() is not None:
15     while True:
16         opc = menu()
17         match opc:
18             case "a":
19                 guardar(registroGrupos(cargar_archivo_json("grupos")), "grupos")
20             case "b":
21                 guardar(registroModulo(cargar_archivo_json("Modulo")), "Modulo")
22             case "c":
23                 guardar(Re_estudiantes(cargar_archivo_json("Estudiantes")), "Estudiantes")
24             case "d":
25                 guardar(registroDocentes(cargar_archivo_json("Docentes")), "Docentes")
26             case "e":
27                 guardar(registroAsistencia(cargar_archivo_json("Estudiantes"), cargar_archivo_json("Modulo"), cargar_archivo_json("asistencia")), "asistencia")
28             case "f":
29                 consultasPorCodigo(cargar_archivo_json("asignacion"), cargar_archivo_json("Estudiantes"))
30             case "g":
31                 cambiarpassword()
32             case "h":
33                 guardar(asignacion(cargar_archivo_json("Estudiantes"), cargar_archivo_json("Modulo"), cargar_archivo_json("grupos"),
34                                     cargar_archivo_json("asignacion")), "asignacion")
35             case "i":
36                 print("Gracias por usar el software")
37                 break
```

Registros

Proporciona
Formato Legible y Estructurado.

Facilidad de Acceso y Manipulación.

Es más eficiente al momento de realizar una consulta.

```
SISGESA > archivo > {} Estudiantes.json > ...
1  {
2      "2233441122": {
3          "codigo": "2233441122",
4          "Nombre": "Andres",
5          "Sexo": "M",
6          "Edad": 16
7      },
8      "1231234567": {
9          "codigo": "1231234567",
10         "Nombre": "laia",
11         "Sexo": "F",
12         "Edad": 17
13     },
14     "3322187432": {
15         "codigo": "3322187432",
16         "Nombre": "Danna",
17         "Sexo": "F",
18         "Edad": 19
19     },
20     "8874632917": {
21         "codigo": "8874632917",
22         "Nombre": "Daniel",
23         "Sexo": "M",
24         "Edad": 18
25     }
26 }
27
```

```
SISGESA > modelo > R_Estudiantes.py > Re_estudiantes
46
47 def Re_estudiantes(datos):
48     print("** Registrar Estudiante **")
49     print("_____")
50     cod = leerCodigo()
51     if cod not in datos:
52         nombre = leerNombre()
53         sexo = leerSexo()
54         edad = leerEdad()
55
56         datEstudiante={
57             "codigo": cod,
58             "Nombre": nombre,
59             "Sexo": sexo,
60             "Edad": edad
61         }
62
63         datos[cod] =datEstudiante
64         print(f"{cod} registrado correctamente.")
65     else:
66         print(f"El código {cod},
67             ya existe en el sistema.")
68
69     return datos
70
```

Persistencia de datos. Cargar

Proporciona

Modularidad.


Fácil de gestionar.

Manejo de errores.

Permite crear un archivo json en caso de no encontrarlo

Flexibilidad.

Permite cargar varios archivos, cambiando el parámetro

SISGESA > persistencia >  cargar.py > ...

```
1  #cargar archivos
2  import json
3
4
5  # Función para cargar archivos JSON dado su nombre y devuelve el contenido
6  #como un diccionario
7  def cargar_archivo_json(nombre_archivo):
8      ruta = f'SISGESA/archivo/{nombre_archivo}.json'
9      try:
10         with open(ruta, 'r') as archivo:
11             return json.load(archivo) #devuelve el contenido del diccionario
12     except ValueError: #manejo de errores
13         print(f"El archivo {nombre_archivo}.json no existe.")
14         return {}# retorna el diccionario vacio en caso de error
15
```

Persistencia de datos. Guardar

SISGESA > persistencia > pesistenciaGuardar.py > ...

```
1 import json
2 #datos es un dict el cual contiene la informacion a guardar
3 #nombre se refiere al "nombre" del archivo json donde posteriormente se guardaran los datos.
4 def guardar(datos, nombre):
5     ruta_archivo = f'SISGESA/archivo/{nombre}.json' #ruta y el nombre del archivo
6
7     # guardar el archivo JSON si existe
8     try:
9         with open(ruta_archivo, 'r') as fd:
10             contenido = json.load(fd) #si existe carga
11             if not isinstance(contenido, dict):
12                 contenido = {}
13     except ValueError: #si no lo encuentra muestra este error
14         contenido = {}
15
16     # Actualizar el contenido con los nuevos datos
17     if datos:
18         contenido.update(datos) # actualizar datos nuevos con existentes
19
20     # Guardar los datos actualizados en el archivo JSON
21     with open(ruta_archivo, 'w') as fd:
22         json.dump(contenido, fd, indent=4)
23
```

Proporciona

Modularidad:

Siendo fácil de gestionar.

Facilidad de Acceso y

Mantenimiento: carga selectiva reduciendo el uso de memoria. es una función reutilizable.

Manejo de errores.

si el archivo json no existe, aloja el contenido en un archivo nuevo y diccionario vacío.

Eficiencia: carga cada registro en su archivo correspondiente.

Asignación

Asignación de estudiantes se asignan a un único grupo y podrán estar matriculados en entre 1 y 3 módulos

Su funcionamiento se asemeja a una tabla relación, donde las Estudiantes, Módulos, y Grupos se toman como entidades

Proporciona.

Integridad Referencial.

Verifica primero si existen los grupos y módulos para asignarlos

Flexibilidad al consultar.

Permite hacer consultas en python mediante diccionarios similares a las SQL, aunque de manera más sencilla y simple

```
SISGESA > modelo > asignacion.py > asignacion
1
2 def buscarGrupo(grupo):
3     codGrupo = input("Ingrese el código del grupo: ")
4     if codGrupo in grupo:
5         return codGrupo
6     else:
7         print("Grupo no existe")
8         return None # Retorna None si el grupo no existe
9
10
11 def asignarModulo(modulo):
12     cont = int(input("¿Cuántos módulos desea ingresar? "))
13     if cont > 3:
14         print("maximo de modulos son 3")
15     else:
16         modulos_asignados = {} # Diccionario para almacenar los módulos
17
18         for i in range(1, cont + 1):
19             codModulo = input(f"Ingrese el código del módulo {i}: ")
20             if codModulo in modulo:
21                 modulos_asignados[f"m{i}"] = codModulo # Asignar el código al diccionario
22             else:
23                 print("Módulo no existe")
24
25         return json.dumps(modulos_asignados) # Retornar el diccionario como JSON
26
27 > def buscarEstudiante(estudiantes):--
28
29
30 def asignacion(estudiantes, modulos, grupos, datos):
31     datosEstudiante = buscarEstudiante(estudiantes)
32     if datosEstudiante is None:
33         return # Salir si el estudiante no existe
34
35     if datosEstudiante not in datos:
36         datosGrupo = buscarGrupo(grupos)
37         if datosGrupo is None:
38             return # Salir si el grupo no existe
39
40         datosAsignacion = {
41             "codigo": datosEstudiante,
42             "Grupo": datosGrupo,
43             "modulo": json.loads(asignarModulo(modulos)) # Convertir el JSON en diccionario
44         }
45         datos[datosEstudiante] = datosAsignacion
46         print(f"Asignación completada para el estudiante {datosEstudiante}.")
47     else:
48         print("Estudiante ya asignado.")
49     return datos # Devolver los datos actualizados
50
51
52
53
54
```


Consultas por código

consulta de:

Consultar los estudiantes matriculados en un grupo.

Consultar los estudiantes inscritos en un módulo.

Proporciona.

Uso Eficiente de Listas.

Manejo de Casos Vacíos.

Flexibilidad en Consultas.

permitiendo decidir al usuario que desea consultar con el sub menú en la función consultas por código.

```
SISGESA > modelo > consultaPorCodigo.py > buscarEstudiantePorModulo
1 def buscarEstudiantePorGrupo(asignacion):
2     codGrupo = input("Ingrese el código del grupo: \n").upper() # Usar input para capturar el valor
3     estudiantes_en_grupo = []
4     for codigo, info in asignacion.items():
5         if info["Grupo"] == codGrupo:
6             estudiantes_en_grupo.append(codigo) # Agrega el estudiante si está en el grupo
7     if not estudiantes_en_grupo: # Si la lista está vacía
8         print("No hay estudiantes en este grupo")
9     return estudiantes_en_grupo
10
11 def consultarEstudiantePorGrupo(asignacion, estudiantes):
12     codEstudiante = buscarEstudiantePorGrupo(asignacion)
13     for codigos in codEstudiante:
14         if codigos in estudiantes:
15             print(f"cod: {codigos} | Nombre: {estudiantes[codigos]['Nombre']}")
16         else:
17             print("No hay estudiantes")
18
19 def buscarEstudiantePorModulo(asignacion):
20     codModulo = input("Ingrese el código del módulo: \n").upper()
21     estudiante_en_modulo = []
22     for codigo, info in asignacion.items():
23         # Verificar si el módulo está en la lista de módulos asignados al estudiante
24         if codModulo in info.get("modulo", {}).values():
25             estudiante_en_modulo.append(codigo) # Agregar el código del estudiante
26
27     if not estudiante_en_modulo: # Si no se encuentra ningún estudiante
28         print("No hay estudiantes asignados a este módulo")
29
30     return estudiante_en_modulo
31
32 def consultarEstudiantePorModulo(asignacion, estudiantes):
33     codEstudiantes = buscarEstudiantePorModulo(asignacion)
34     for codigo in codEstudiantes:
35         if codigo in estudiantes:
36             print(f"Nombre del estudiante: {estudiantes[codigo]['Nombre']}")
37         else:
38             print(f"No se encontró información del estudiante con código {codigo}")
39
40 def consultasPorCodigo(asignacion, estudiantes):
41     while True:
42         op = int(input("Consultas:\n\t1. Estudiantes por Grupo:\n\t2. Estudiantes por Modulo\n\t3. Salir\n"))
43         if op == 1:
44             consultarEstudiantePorGrupo(asignacion, estudiantes)
45         elif op == 2:
46             consultarEstudiantePorModulo(asignacion, estudiantes)
47         elif op == 3:
48             break
```

Validación Asistencia

```
SISESA > modelo > H_asistencia.py > registroAsistencia
1 from datetime import datetime
2 import json
3
4 def leerCodigoEstudiante():
5     while True:
6         try:
7             cod = input("Ingrese el código: \n")
8             if len(cod.strip()) == 0:
9                 print("Error. Código inválido")
10                continue
11            return cod
12        except Exception as e:
13            print("Error al ingresar el código. \n" + str(e))
14
15 def leerCodigoModulo():
16     while True:
17         try:
18             cod = input("Ingrese el código: \n")
19             if len(cod.strip()) == 0:
20                 print("Error. Código inválido")
21                 continue
22             return cod
23        except Exception as e:
24            print("Error al ingresar el código. \n" + str(e))
25
26 def leerFechaEntrada():
27     # Retorna la fecha actual como una cadena en formato DD/MM/YYYY
28     return datetime.now().strftime("%d/%m/%Y %H:%M")
29
30
31 def registroAsistencia(estudiantes, modulos, datos):
32     print("***Registrar**")
33     print("_____")
34     codEstudiante = leerCodigoEstudiante()
35     codModulo = leerCodigoModulo()
36
37     if codEstudiante in estudiantes and codModulo in modulos:
38         fechaEntrada = leerFechaEntrada() # Fecha actual como cadena
39
40         # Crear el nuevo registro como un diccionario
41         modelDat = {
42             "codigoEstudiante": codEstudiante,
43             "codigoModulo": codModulo,
44             "fechaEntrada": fechaEntrada,
45         }
46
47         # Agregar el nuevo registro a 'datos', utilizando el código como clave
48         datos[cod] = modelDat
49         print(f"{cod} registrado correctamente.")
50     else:
51         print(f"El código {codEstudiante} o {codModulo} no existe en el sistema.")
52     return datos
```

Se realiza el ingreso de asistencia en la que se toma en cuenta la hora actual, por lo que se emplea la función datetime

```
31 def registroAsistencia(estudiantes, modulos, datos):
32     print("***Registrar**")
33     print("_____")
34     codEstudiante = leerCodigoEstudiante()
35     codModulo = leerCodigoModulo()
36
37     if codEstudiante in estudiantes and codModulo in modulos:
38         fechaEntrada = leerFechaEntrada() # Fecha actual como cadena
39
40         # Crear el nuevo registro como un diccionario
41         modelDat = {
42             "codigoEstudiante": codEstudiante,
43             "codigoModulo": codModulo,
44             "fechaEntrada": fechaEntrada,
45         }
46
47         cod = len(datos) + 1 # el contador es la consulta
48         # Agregar el nuevo registro a 'datos', utilizando el código como clave
49         datos[cod] = modelDat
50         print(f"{cod} registrado correctamente.")
51     else:
52         print(f"El código {codEstudiante} o {codModulo} no existe en el sistema.")
53
54     return datos
```

Gracias por ver!