```python
    def getCreditos(self):
        """
        Devuelve la cantidad de créditos de la materia.

        Returns:
            int: Créditos de la materia.
        """
        if self.infoMateria == None:
            raise TypeError("...")
        return self.infoMateria[2]


class GestorExpediente:
    """
    Crea, carga y modifica archivos JSON para cada estudiante.
    Permite almacenar y gestionar sus datos.
    """

    def __init__(self, nombre: str, ci: int, añoIngreso: int):
        """
        Inicializa la clase con los datos del estudiante.

        Args:
            nombre (str): Nombre del estudiante.
            ci (int): Cédula de identidad del estudiante.
            añoIngreso (int): Año de ingreso a la carrera.
        """
        self.datos = {"Nombre": nombre,
                      "Cedula": ci,
                      "Año Ingreso": añoIngreso,
                      "Materias Aprobadas": [],
                      "Materias Matriculadas": [],
                      "Inscripción Examen": [],
                      "Créditos": 0}

        nombre_archivo = nombre.replace(" ", "_")
        self.archivo = f"{nombre_archivo}_{ci}.json"

    def crearArchivo(self, indentacion: int = 4):
        """
        Crea un archivo JSON para el estudiante con datos iniciales.

        Args:
            indentacion (int): Nivel de indentación para el formato JSON.
        """
        with open(self.archivo, 'w', encoding='utf-8') as file:
            json.dump(self.datos, file, ensure_ascii=False, indent=i

        self.data = self.datos
```

Ingeniería Biomédica

UTEC
Universidad Tecnológica

SISTEMA DE MATRICULACIONES

Estudiantes: Juan Acosta, Joaquin Sanchez y Sosa Florencia

Programación Digital Avanzada

Docentes: MSc. Bioing. Baldezzari Lucas e Ing. Alvarez Emilian

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self,
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
        self.actualizar_archi
```

INTRODUCCION

- En la carrera de Ingeniería Biomédica de la UTEC, actualmente el sistema de inscripciones no verifica automáticamente si el estudiante cumple con las materias previas requeridas.
- Esto genera errores frecuentes y una sobrecarga para la secretaría.
- Este trabajo propone un Sistema de Matriculación (SdM) desarrollado en Python, que valida automáticamente las previas antes de permitir la inscripción a materias o exámenes, aplicando conceptos de Programación Orientada a Objetos (POO).

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self,
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
        self.actualizar_archi
```

OBJETIVOS

Objetivo General

- Implementar un Sistema de Matriculaciones -básico- a Unidades Curriculares y Exámenes para la carrera de Ingeniería Biomédica (plan 2021) utilizando Python y los conceptos de Programación Orientada a Objetos (POO) vistos hasta el momento.

Objetivos Específicos

- Reforzar los conceptos de Programación Orientada a Objetos (POO), incluyendo abstracción, encapsulamiento y los distintos tipos de relaciones entre clases, como asociación simple, agregación y composición.
- Reforzar los conceptos de Clase y Objeto mediante la implementación del sistema de matriculaciones ya que deberá definir clases, objetos e interacciones entre estos.
- Reforzar los conceptos de diagramas UML.
- Implementar conceptos de Manejo de Excepciones y Manejo de Archivos en Python

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self,
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
        self.actualizar_archi
```

IMPLEMENTACIÓN

- **PlanEstudio:** encargada de cargar el plan desde un archivo JSON y brindar acceso a la información de cada unidad curricular.
- **UC:** permite consultar los datos de cada materia y validar requisitos.
- **GestorExpediente:** gestiona el archivo JSON de cada estudiante y el archivo global de registros.
- **Clase Estudiante:** permite realizar inscripciones y matriculaciones
- **Clase Secretaria:** cumple funciones administrativas más amplias, como modificar aprobaciones o visualizar inscriptos por materia.

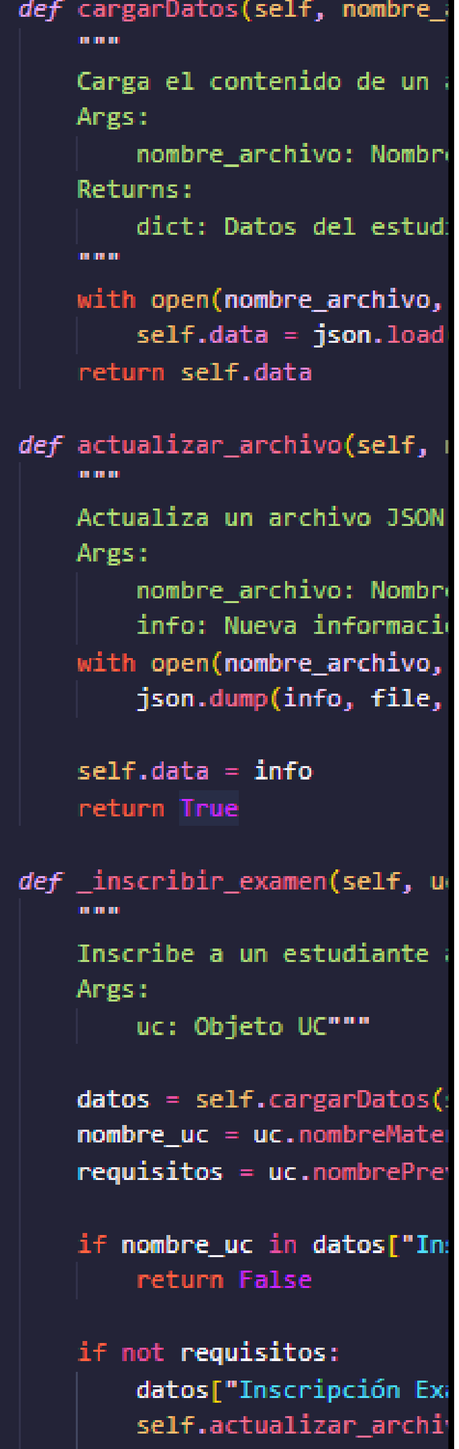
```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self,
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
        self.actualizar_archi
```

CLASIFICACIÓN

Clases estructurales (modelo de datos)

Son las que representan la base académica sobre la cual se valida el sistema:

- **PlanEstudio:** Carga y organiza el plan de estudios.
- **UC (Unidad Curricular):** Representa una materia específica, con sus previas y créditos.



```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self,
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
        self.actualizar_archi
```

CLASIFICACIÓN

Clases de gestión y lógica del sistema

Esta clases administran datos, validaciones y operaciones internas:

- **GestorExpediente:** Maneja los archivos JSON de cada estudiante y los registros globales. Es el núcleo de la lógica operativa.

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self,
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
        self.actualizar_archi
```

CLASIFICACIÓN

Clases de usuario (interfaz con el sistema)

Modelan los roles desde los cuales se usan las funcionalidades:

- **Estudiante:** Permite realizar acciones como inscribirse, consultar su estado y matriculase
- **Secretaria:** Simula el rol administrativo, con más permisos y acceso a múltiples funciones

DIAGRAMA UML

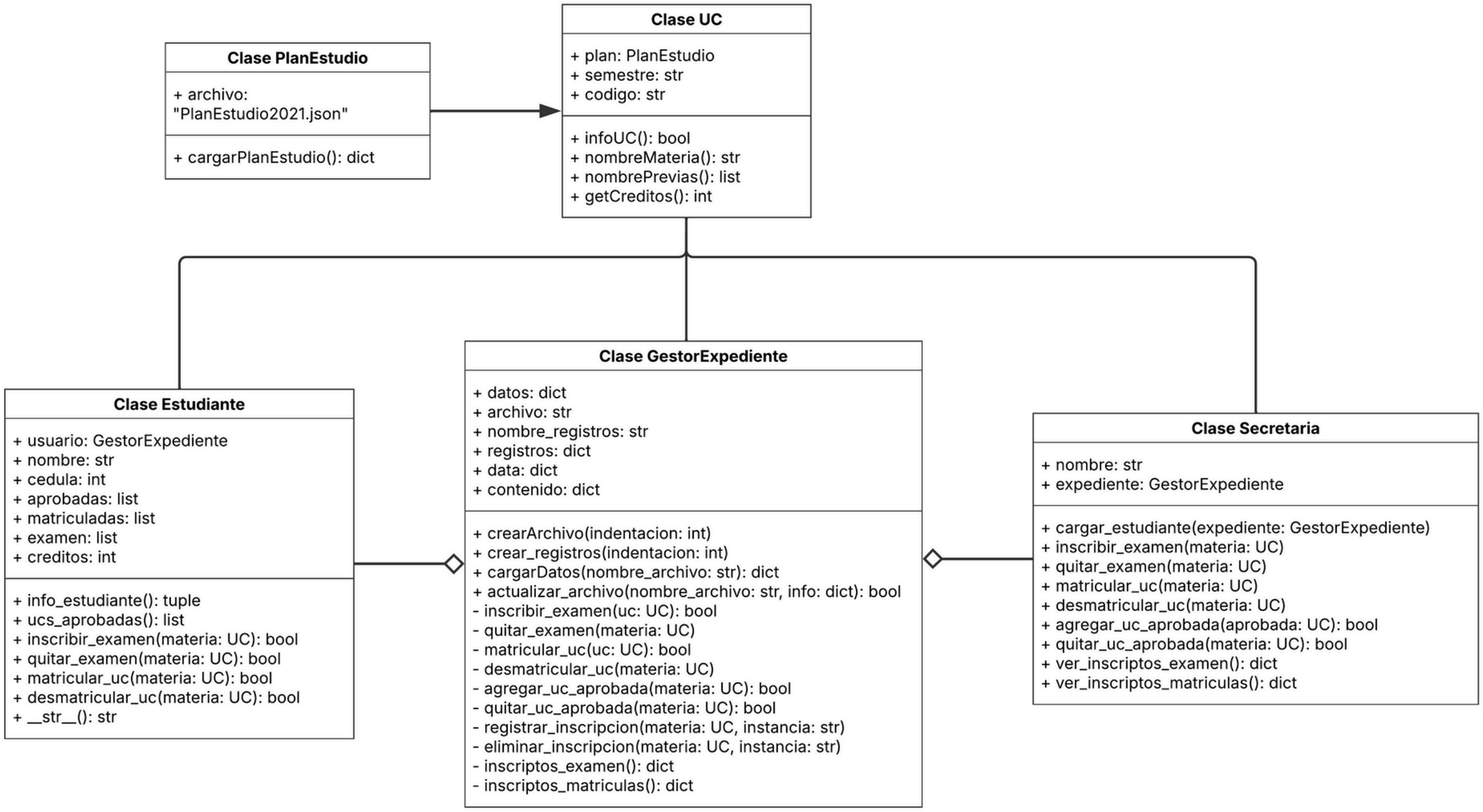


Figura 1. Diagrama UML

```
def cargarDatos(self, nombre
    """
    Carga el contenido de un
    Args:
        nombre_archivo: Nomb
    Returns:
        dict: Datos del estu
    """
    with open(nombre_archivo
        self.data = json.loa
    return self.data

def actualizar_archivo(self,
    """
    Actualiza un archivo JSO
    Args:
        nombre_archivo: Nomb
        info: Nueva informac
    with open(nombre_archivo
        json.dump(info, file

    self.data = info
    return True

def _inscribir_examen(self,
    """
    Inscribe a un estudiante
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos
    nombre_uc = uc.nombreMat
    requisitos = uc.nombrePr

    if nombre_uc in datos["I
        return False

    if not requisitos:
        datos["Inscripción E
        self.actualizar_arch
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

1. Importar las clases necesarias

Se importa el archivo SistemaMatriculaciones.py y las clases que se usarán en el sistema.

```
from SistemaMatriculaciones import UC, PlanEstudio, Estudiante, Secretaria, GestorExpediente
```

2. Cargar el plan de estudios

Se crea un objeto del plan y se carga el archivo PlanEstudio2021.json, que contiene la información de todas las materias, sus previas y créditos.

```
planIBIO = PlanEstudio("PlanEstudio2021.json")
planIBIO.cargarPlanEstudio()
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

3. Instanciar una unidad curricular

Se selecciona una materia específica del plan para operar con ella, por ejemplo Matemática del semestre 1.

```
matematica = UC(planIBIO, "Semestre 1", "S1UC1", )
print(matematica.infoUC())
print(matematica.nombreMateria())
print(matematica.nombrePrevias())
print(matematica.getCreditos())
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

4. Crear el expediente del estudiante

Se genera el expediente académico del estudiante en un archivo .json, junto con el archivo Registros.json que guarda todas las inscripciones del sistema.

```
expediente = GestorExpediente("Agustín Acosta", 56692489, 2022)
expediente.crearArchivo()
expediente.crear_registros()
```

5. Se Instancia al estudiante

Se crea un objeto Estudiante a partir del expediente, lo que permite acceder y gestionar su información académica.

```
agustin = Estudiante(expediente)
print(agustin)
```

→

```
Información del estudiante:
Nombre: Agustín Acosta
C.I: 56692489
Materias aprobadas: []
Materias Matriculadas: []
Inscripcion Examenes: []
Créditos: 0
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

4. Crear el expediente del estudiante

Se genera el expediente académico del estudiante en un archivo .json, junto con el archivo Registros.json que guarda todas las inscripciones del sistema.

```
expediente = GestorExpediente("Agustín Acosta", 56692489, 2022)
expediente.crearArchivo()
expediente.crear_registros()
```

5. Se Instancia al estudiante

Se crea un objeto Estudiante a partir del expediente, lo que permite acceder y gestionar su información académica.

```
agustin = Estudiante(expediente)
print(agustin)
```

→

```
Información del estudiante:
Nombre: Agustín Acosta
C.I: 56692489
Materias aprobadas: []
Materias Matriculadas: []
Inscripcion Examenes: []
Créditos: 0
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

Operaciones desde el rol estudiante

1. Matricularse

```
agustin.matricular_uc(matematica)
agustin=Estudiante(expediente)
print(agustin)
```

```
Información del estudiante:
Nombre: Agustín Acosta
C.I: 56692489
Materias aprobadas: []
Materias Matriculadas: ['Álgebra, Análisis y Geometría Analítica']
Inscripcion Examenes: []
Créditos: 0
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

Operaciones desde el rol estudiante

2. Desmatricularse

```
agustin.desmatricular_uc(matematica)
agustin=Estudiante(expediente)
print(agustin)
```

```
Información del estudiante:
Nombre: Agustín Acosta
C.I: 56692489
Materias aprobadas: []
Materias Matriculadas: []
Inscripcion Examenes: []
Créditos: 0
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

Operaciones desde el rol estudiante

3. Inscribirse a un Examen

```
agustin.inscribir_examen(matematica)
agustin=Estudiante(expediente)
print(agustin)
```

```
Información del estudiante:
Nombre: Agustín Acosta
C.I: 56692489
Materias aprobadas: []
Materias Matriculadas: []
Inscripcion Examenes: ['Álgebra, Análisis y Geometría Analítica']
Créditos: 0
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

Operaciones desde el rol estudiante

3. Darse de baja en un Examen

```
agustin.quitar_examen(matematica)
agustin=Estudiante(expediente)
print(agustin)
```

```
Información del estudiante:
Nombre: Agustín Acosta
C.I: 56692489
Materias aprobadas: []
Materias Matriculadas: []
Inscripcion Examenes: []
Créditos: 0
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

Operaciones desde el rol de la secretaria

1. Se crea un objeto Secretaria para realizar acciones administrativas sobre los expedientes académicos.

```
jessica = Secretaria("Jessica")
```

2. La secretaria accede al expediente del estudiante para gestionar inscripciones, matrículas y materias aprobadas.

```
jessica.cargar_estudiante(expediente)
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

Operaciones desde el rol de la secretaria

3. Matricular al Estudiante

```
jessica.matricular_uc(fisica)
estudiante = Estudiante(jessica.expediente)
print(estudiante)
```

```
Información del estudiante:
Nombre: Agustín Acosta
C.I: 56692489
Materias aprobadas: []
Materias Matriculadas: ['Mecánica, Ondas y Calor']
Inscripcion Examenes: []
Créditos: 0
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

Operaciones desde el rol de la secretaria

4. Desmatricular al Estudiante

```
jessica.desmatricular_uc(fisica)
estudiante=Estudiante(jessica.expediente)
print(estudiante)
```

```
Información del estudiante:
Nombre: Agustín Acosta
C.I: 56692489
Materias aprobadas: []
Materias Matriculadas: []
Inscripcion Examenes: []
Créditos: 0
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

Operaciones desde el rol de la secretaria

5. Incribir a un examen a un estudiante

```
jessica.inscribir_examen(fisica)
estudiante=Estudiante(jessica.expediente)
print(estudiante)
```

```
Información del estudiante:
Nombre: Agustín Acosta
C.I: 56692489
Materias aprobadas: []
Materias Matriculadas: []
Inscripcion Examenes: ['Mecánica, Ondas y Calor']
Créditos: 0
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

Operaciones desde el rol de la secretaria

6. Dar de baja de un examen a un estudiante

```
jessica.quitar_examen(fisica)
estudiante=Estudiante(jessica.expediente)
print(estudiante)
```

```
Información del estudiante:
Nombre: Agustín Acosta
C.I: 56692489
Materias aprobadas: []
Materias Matriculadas: []
Inscripcion Examenes: []
Créditos: 0
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

Operaciones desde el rol de la secretaria

7. Se añade la UC como aprobada y se suman los créditos

```
jessica.agregar_uc_aprobada(matematica)
estudiante=Estudiante(jessica.expediente)
print(estudiante)
```

```
Información del estudiante:
Nombre: Agustín Acosta
C.I: 56692489
Materias aprobadas: ['Álgebra, Análisis y Geometría Analítica']
Materias Matriculadas: []
Inscripcion Examenes: []
Créditos: 8
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

Operaciones desde el rol de la secretaria

8. Se elimina la UC aprobada y se descuentan los créditos

```
jessica.quitar_uc_aprobada(matematica)
estudiante=Estudiante(jessica.expediente)
print(estudiante)
```

```
Información del estudiante:
Nombre: Agustín Acosta
C.I: 56692489
Materias aprobadas: []
Materias Matriculadas: ['Óptica y Radiaciones']
Inscripcion Examenes: ['Óptica y Radiaciones']
Créditos: 0
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

Operaciones desde el rol de la secretaria

9. La secretaria visualiza las listas de estudiantes inscriptos a exámenes o materias mediante los métodos correspondientes.

```
jessica.ver_inscriptos_matriculas()
```

```
{'Taller Inicial de Tecnologías': ['Florencia Sosa', 'Agustín Acosta'],
 'Álgebra, Análisis y Geometría Analítica': ['Florencia Sosa']}
```

```
jessica.ver_inscriptos_examen()
```

```
{'Álgebra, Análisis y Geometría Analítica': ['Florencia Sosa'],
 'Taller Inicial de Tecnologías': ['Florencia Sosa']}
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

Operaciones desde el rol de la secretaria

10. Ni la secretaria ni el estudiante pueden inscribirse a exámenes o materias si no se cumplen las previas. Sin embargo, si la secretaria registra las materias previas como aprobadas, la inscripción queda habilitada.

```
jessica.agregar_uc_aprobada(matematica)
jessica.agregar_uc_aprobada(fisica)
estudiante=Estudiante(jessica.expediente)
print(estudiante)
```

```
Información del estudiante:
Nombre: Agustín Acosta
C.I: 56692489
Materias aprobadas: ['Álgebra, Análisis y Geometría Analítica', 'Mecánica, Ondas y Calor']
Materias Matriculadas: []
Inscripcion Examenes: []
Créditos: 18
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

Operaciones desde el rol de la secretaria

10. Ni la secretaria ni el estudiante pueden inscribirse a exámenes o materias si no se cumplen las previas. Sin embargo, si la secretaria registra las materias previas como aprobadas, la inscripción queda habilitada.

```
jessica.matricular_uc(optica)
estudiante=Estudiante(jessica.expediente)
print(estudiante)
```

```
Información del estudiante:
Nombre: Agustín Acosta
C.I: 56692489
Materias aprobadas: ['Álgebra, Análisis y Geometría Analítica', 'Mecánica, Ondas y Calor']
Materias Matriculadas: ['Óptica y Radiaciones']
Inscripcion Examenes: []
Créditos: 18
```

```
def cargarDatos(self, nombre_
    """
    Carga el contenido de un 
    Args:
        nombre_archivo: Nombr
    Returns:
        dict: Datos del estud
    """
    with open(nombre_archivo,
        self.data = json.load
    return self.data

def actualizar_archivo(self, 
    """
    Actualiza un archivo JSON
    Args:
        nombre_archivo: Nombr
        info: Nueva informaci
    with open(nombre_archivo,
        json.dump(info, file,

    self.data = info
    return True

def _inscribir_examen(self, u
    """
    Inscribe a un estudiante 
    Args:
        uc: Objeto UC"""

    datos = self.cargarDatos(
    nombre_uc = uc.nombreMate
    requisitos = uc.nombrePre

    if nombre_uc in datos["In
        return False

    if not requisitos:
        datos["Inscripción Ex
```

MODO DE USO

Operaciones desde el rol de la secretaria

10. Ni la secretaria ni el estudiante pueden inscribirse a exámenes o materias si no se cumplen las previas. Sin embargo, si la secretaria registra las materias previas como aprobadas, la inscripción queda habilitada.

```
jessica.inscribir_examen(optica)
estudiante=Estudiante(jessica.expediente)
print(estudiante)
```

```
Información del estudiante:
Nombre: Agustín Acosta
C.I: 56692489
Materias aprobadas: ['Álgebra, Análisis y Geometría Analítica', 'Mecánica, Ondas y Calor']
Materias Matriculadas: ['Óptica y Radiaciones']
Inscripcion Examenes: ['Óptica y Radiaciones']
Créditos: 18
```

```
        self.data = info
        return True

    def _inscribir_examen(self, uc: UC):
        """
```

CONCLUSION

Se logró desarrollar un sistema funcional que permite gestionar inscripciones a materias y exámenes respetando las previas definidas en el plan de estudios.
Durante su implementación se aplicaron conceptos clave de Programación Orientada a Objetos y manejo de archivos JSON para registrar el historial académico de los estudiantes.
El sistema distingue dos roles: estudiante y secretaría, ambos con validaciones automáticas de requisitos.

```
        self.data = info
        return True

    def _inscribir_examen(self, uc: UC):
        """
```

¡MUCHAS GRACIAS!