PROYECTO FINAL MD1

SIGVAL

Realizado por: Melissa Espitia - 2359439-2724

Docente: Luis Germán Toro Pareja

UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
SECCIONAL TULUA
2025

Introducción

El proyecto **SIGVAL.py** es un sistema de gestión universitaria desarrollado en Python, que permite administrar estudiantes, profesores, cursos, horarios y matrículas. El sistema utiliza estructuras de datos basadas en conjuntos para garantizar la unicidad de los registros y facilitar operaciones de alta, baja y consulta. A continuación, se presenta una descripción detallada de la estructura, funcionamiento y conclusiones del sistema.

Estructura y Modelado Matemático

El sistema modela los elementos principales de la universidad como conjuntos:

- **Estudiantes:**

$$S = \{(c, n) \mid c \in \text{Códigos de estudiante}, n \in \text{Nombres}\}$$

- **Profesores:**

$$P = \{(i, n) \mid i \in \text{IDs de profesor}, n \in \text{Nombres}\}$$

- **Cursos:**

$$C = \{(id, n, g) \mid id \in \text{IDs de curso}, n \in \text{Nombres}, g \in \text{Grupos}\}$$

- **Horarios de cursos:**

$$H = \{(id, g, t) \mid id \in \text{IDs de curso}, g \in \text{Grupos}, t \in \text{Bloques de tiempo}\}$$

- **Asignación de profesores a cursos:**

$$A = \{(id, g, i) \mid id \in \text{IDs de curso}, g \in \text{Grupos}, i \in \text{IDs de profesor}\}$$

- **Matrículas de estudiantes:**

$$M = \{(c, id, g) \mid c \in \text{Códigos de estudiante}, id \in \text{IDs de curso}, g \in \text{Grupos}\}$$

Cada función del sistema corresponde a una operación sobre estos conjuntos, asegurando la integridad de los datos y evitando duplicados mediante la propiedad de unicidad de los conjuntos.

Funciones Principales

1. Registro de Estudiantes y Profesores

- `reg_student(student_code, student_name)`
  Añade un estudiante al conjunto S si no existe ya un estudiante con el mismo código c. Matemáticamente:

$$(c, n) \notin S \implies S := S \cup \{(c, n)\}$$

- `add_professor(professor_id, professor_name)`
  Añade un profesor al conjunto P si el ID no está registrado.

2. Registro de Cursos

- `add_course(course_id, course_name, group)`
  Añade un curso al conjunto C si no existe ya un curso con el mismo ID y grupo.

$$(id, *, g) \notin C \implies C := C \cup \{(id, n, g)\}$$

3. Matrícula y Asignación

- `enroll_student_in_course(student_code, course_id, group)`
  Matricula a un estudiante en un curso-grupo, verificando que no haya conflicto de horarios.

$$\text{Si } \forall (id', g') \in M_c, H_{id,g} \cap H_{id',g'} = \emptyset \implies M := M \cup \{(c, id, g)\}$$

  Donde M_c es el conjunto de cursos en los que está inscrito el estudiante c.

- `assign_professor_to_course(course_id, group, professor_id)`
  Asigna un profesor a un curso-grupo, también verificando conflictos de horario.

4. Eliminación y Consulta

- `remove_student`, `remove_professor`, `remove_course`
  Eliminan elementos de los conjuntos y sus relaciones asociadas, garantizando la consistencia de los datos.

- `view_courses_and_enrollments()`
  Muestra todos los cursos, los estudiantes inscritos y los profesores asignados, recorriendo los conjuntos y mostrando la información relacionada.

Conclusiones

1. **Integridad de Datos:** El uso de conjuntos garantiza que no existan duplicados en los registros de estudiantes, profesores y cursos, cumpliendo con la restricción de unicidad de los identificadores.

2. **Relaciones y Consistencia:** Las funciones de eliminación aseguran que, al borrar un estudiante, profesor o curso, se eliminan también todas las relaciones asociadas, manteniendo la consistencia referencial entre los conjuntos.

3. **Prevención de Conflictos:** El sistema implementa validaciones para evitar conflictos de horarios tanto en la matrícula de estudiantes como en la asignación de profesores, utilizando la intersección de conjuntos de horarios.

4. **Escalabilidad y Simplicidad:** La estructura basada en conjuntos y operaciones de teoría de conjuntos permite que el sistema sea fácilmente escalable y comprensible, facilitando futuras ampliaciones o modificaciones.

5. **Modularidad:** Cada función realiza una tarea específica sobre los conjuntos, lo que mejora la mantenibilidad y la claridad del código.

Recomendaciones

- Se podría mejorar la interfaz de usuario para hacerla más amigable.
- Implementar persistencia de datos para no perder la información al cerrar el programa.
- Añadir validaciones adicionales para los formatos de entrada.

SIGVAL.py es un ejemplo claro de cómo la teoría de conjuntos y el modelado matemático pueden aplicarse eficazmente en el desarrollo de sistemas de gestión académica.