Universidad Tecnológica

Trabajo práctico - Módulos 3 y 4

Sistema de matriculaciones para Ingeniería Biomédica

Programación Digital Avanzada

Carrera: Ingeniería Biomédica
Semestre: 7
Profesor: MSc. Bioing. BALDEZZARI Lucas

2022

1 Introducción

Actualmente en las carreras de la Universidad Tecnológica (UTEC), cuando un/una estudiante desea puede matricularse a una *Unidad Curricular* (UC) o inscribirse a un *Examen Final* (EF) a través del sistema *Universita XXI*. Al momento, este sistema no tiene en cuenta las previaturas que exige el plan de estudio de una carrera, siendo tarea del o la estudiante *chequear* si cumple las condiciones antes de matricularse o inscribirse. Sin embargo, en ocasiones sucede que la persona se matricula a una UC o se inscribe a un examen y el sistema no pone ninguna restricción, y es tarea de la secretaria o el secretario de carrera corroborar que la persona inscripta cumple con los requisitos.

Esto último es muy ineficiente, ya que en el caso de que una o un estudiante se inscriba a una UC, sin cumplir con las condiciones que exige el plan de estudios, trae aparejado que la secretaria/o de carrera deba corroborar, estudiante por estudiante, si todos y todas están correctamente inscriptos/as. Por lo tanto, esto provoca que deba darse de baja a la persona que se ha inscripto erróneamente y en ocasiones se lo da de baja hasta tres o cuatro semanas luego de iniciadas las clases, lo cual es una perdida de tiempo para el o la estudiante, para el cuerpo docente y también para el departamento de secretaría/coordinación que debe hacer una terea que se supone, debe estar automatizada desde un inicio por el mismo sistema de matriculaciones de la universidad.

Con este laboratorio se propone la implementación, básica, de un Sistema de Matriculaciones (SdM) a UCs y exámenes que considere las UCs previas que son requisitos para la matriculación/inscripción según el plan de estudios de la carrera de *Ingeniería Biomédica* (IBIO) del año 2019, esto quiere decir que el sistema deberá considerar para una UC en particular, cuales son las UCs que deben estar salvadas/exoneradas y aquellas que pueden haber quedado a examen para que un estudiante pueda cursar o tener derecho a examen.

2 Objetivos

2.1 Objetivo general

Implementar un Sistema básico de Matriculaciones a Unidades Curriculares y Exámenes para la carrera de Ingeniería Biomédica (plan 2019) utilizando Python.

2.2 Objetivos específicos

- Reforzar los conceptos de Programación Orientada a Objetos (POO), *Herencia*, *Abstracción*, *Encapsulamiento*.
- 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.

Además se espera que el/la estudiante pueda mejorar el pensamiento lógico a través de la resolución del problema planteado.

3 Materiales y Métodos

Para llevar a cabo este trabajo práctico necesitará.

- Python 3.9 (o superior).

- Visual Studio (o algún otro entorno de desarrollo).
- Un repositorio en Github.com a donde almacenar la implementación y toda la documentación generada.
- Plan de estudios de la carrera de Ingeniería Biomédica.

Se recomienda que la codificación para llevar a cabo el *SdM*, es decir, las clases con sus atributos y métodos, como así también los objetos creados y sus interacciones, sean llevadas a cabo utilizando un entorno de desarrollo adecuado. No se recomienda el uso de Jupyter Notebook debido a que no posee las fortalezas (autocompletar, depuración de código, entre otras) que sí posee un entorno de desarrollo como puede ser Visual Studio.

Todo el material codificado, junto con la documentación generada, deberá estar sincronizada en el repositorio de Github.

El trabajo podrá realizarse en grupo, hasta un máximo de 3 personas.

4 Requisitos del Sistema de Matriculaciones

Sabemos que el SdM posee diferentes actores/as, como ser, estudiantes, secretarias/os, docentes, etc. Cada uno/a de estos/as actores posee diferentes roles dentro de la institución y a su vez posee diferentes *permisos* en lo que al *SdM* respecta.

Es importante que realice un análisis de las personas que son partícipes dentro del *SdM* y cómo interactúan entre sí, para así determinar las *Clases*, y en base a esto, poder definir qué atributos y qué métodos tendrán para cumplir con los objetivos del trabajo.

Por ejemplo, ¿qué información se necesita de un estudiante para llevar a cabo el *SdM*? Por mencionar algunos, necesitará de un Nombre, Apellido, Cédula, UCs aprobadas y UCs previas, entre otras. Aquí deberá ser **crítico/a** para determinar cuanta información de estudiante es necesaria. Esto último es una clara aplicación de *Abstracción*. Recuerde que *menos es más*.

Considerando esto, el *SdM* deberá tener, cómo mínimo, las siguientes características:

Para los y las estudiantes

- Permitir la inscripción a UCs y/o Exámenes de la carrera de IBIO. Solamente podrán matricularse y/o inscribirse aquellas personas que cumplan con las previaturas según el plan de estudio.
- Un estudiante podrá inscribirse a más de un curso/UC y/o exámen siempre y cuando cumpla con las previaturas.

Para secretaria/o de carrera

- Permitir que pueda matricular o desmatricular a un/a estudiante a una UC o bien a un exámen.
- Deberá poder acceder a la lista de inscriptos/as a un determinado curso/UC y/o exámen.

Se supone que una persona que no cumpla con las previaturas no debería ser matriculada, ni siquiera por la secretaria. No obstante, existen casos en donde una persona podría querer inscribirse fuera de la ventana de tiempo estipulada por UTEC y si posee una adecuada justificación, desde secretaría se la podría inscribir.

Algo similar aplica para la desmatriculación.

Para coordinadora/or de carrera

- Deberá poder acceder a la lista de inscriptos/as a un determinado curso/UC y/o exámen.

Del *SdM* en general

Su programa debe permitir,

- Leer/Escribir un archivo de texto los/as estudiantes matriculados/as a una UC/curso en particular.
- Leer/Escribir en un archivo de texto los/as estudiantes inscriptos/as a un exámen en particular.

Esto último puede ser útil para formar las actas, listas de asistencia, etc.

Para lograr la gestión de archivos puede pensar en un Módulo llamado *GestorDeDatos*. Éste módulo podría tener funciones que permitan leer y escribir los archivos mencionados anteriormente, similar a un gestor de base de datos.

Importante: Cada una de las clases de su programa deberá tener, como mínimo,

- Un constructor adecuado.
- Atributos públicos y privados. Utilice su criterio para definir que atributos serán públicos y cuales privados. En los casos que sea necesario acceder y/o modificar atributos privados de alguna clase, tenga en cuenta de utilizad decoradores como getters y setters para no acceder a estos atributos de una manera directa.
- Métodos públicos y privados. Utilice su criterio para definir que métodos serán públicos y cuales privados.

Puede proponer la cantidad de atributos y métodos que crea necesario.

5 Entregables

Se debe entregar,

- Un diagrama UML de las clases que utiliza su implementación.
- Todos los archivos **.py* y **.txt* que haya implementado, en la misma estructura de directorios con la que diseño su programa.
- Una Jupyter Notebook demostrando el funcionamiento de su implementación.
- Una lista (en un archivo *txt*) de las librerías de terceros que se necesitan para la implementación de su solución.

Cada trabajo deberá ser expuesto, de manera oral, frente al docente y sus compañeros/as.

6 Ejemplo de implementación

A continuación se deja un ejemplo de implementación. El mismo es una idea que será discutida en clase. Puede usarse esto como referencia para realizar el SdM o bien implementar uno diferente, lo importante es poder resolver lo que se pide.

```
## Trabajo práctico - Modulos 3 y 4
## Autor: Baldezzari, Lucas
## UC: PDA
## Año: 2022

## EJEMPLO de implementación

from sistemaMatricu import Estudiante, Secretaria, Curso, Examen
from uc import UC #importamos la clase UC. Servirá para crear UCs.
from planDeEstudio import IBIO2018 #importamos el plan de estudio de IBIO

## ******* Definimos algunos objetos Estudiante *******
estudiante1 = Estudiante("John", "Constantine", CI = 25554444, ingreso = 2022) #genero un
                                                    estudiante
estudiante2 = Estudiante("Albert", "Einstein", CI = 38885552, ingreso = 2022) #genero otro
                                                    estudiante
estudiante3 = Estudiante("Isaac", "Asimov", CI = 16665553, ingreso = 2019) #genero otro
                                                    estudiante
estudiante4 = Estudiante("Marie", "Curi", CI = 16668883, ingreso = 2021) #genero otro
                                                    estudiante
secretaria = Secretaria("Virginia", "Catala") #creamos una secretaria

####### Comentarios ******
#1) Supongamos que cada estudiante tiene un método para saber a que UCs esta inscripto
#2) Supongamos que cada estudiante tiene un método para saber que UCs tiene aprobada
#3) Supongamos que cada estudainte tiene un método para saber que UCs tiene regular o sin
                                                    aprobar.
#4) Supongamos que los/as estudiantes que ingresaron en el 2022 aún no tienen
#ninguna UC aprobada pero sí estan cursando todas las del primer semestre.
#5) Supongamos que los/as estudiantes que ingresaron antes del 2022
#tienen UCs aprobadas y regulares.
#6) Recuerde que esto es una implementación de ejemplo.
#Usted podrá crear métodos diferentes y/o agregar nuevos.

#Veamos que UCs esta cursando el estudiante1
print(estudiante1.ucsCursando())
#Este método pordía devolver una lista similar a: ['S1UC1', 'S1UC2', 'S1UC3', 'S1UC4', 'S1UC5
                                                    ', 'S1UC6', 'S1UC7']
"""En este ejemplo, la lista anterior devuelve los códigos de las UCs del primer semestre.

Ejemplo:

- S1UC1 sería "Álgebra, Análisis y Geometría Analítica" o
- S1UC2 = "Mecánica, Ondas y Calor"

La codificación propuesta es:
Semestre-Número-UC-Posición en el plan de estudio

Usted podría definir otra codificación considerando la carrera, el año del plan de estudio,
                                                    etc.
"""

#Veamos que UCs aprobadas tiene el estudiante2
print(estudiante2.ucsAprobadas())
#Esto podría devolver None, ya que el estudiante2 ha ingresado en el 2022.

#No obstante el estudiante4 que ingreso en 2021 podría tener algunas UCs aprobadas, veamos.
print(estudiante4.ucsAprobadas())
#podríamos ver algo como: ['S1UC1','S1UC2','S2UC2']

"""
*******************************************************
          Objeto UC
*******************************************************

Pude pensar en diseñar una clase UC.
Esta clase representará a una UC en particular de la carrera. Puede pensar en atributos como,
nombre, semestre, previaturas (regulares y aprobadas) necesarias para cursar la UC, créditos,
carga horaria, etc.

Puede pensar en métodos para que la clase UC le devuelva cuales son las previaturas que
necesita, o en que semestre se cursa.
```

```
"""

## Creamos la UC Álgebra, Análisis y Geometría Analítica del primer semestre usando
## la codificación mencionada anteriormente.

S1UC1 = UC(nombre = "Álgebra, Análisis y Geometría Analítica", code = 'S1UC1',
            ucregulares = None, ucaprobadas = None,
            semestre = 1, créditos = 8)

## Creamos la UC Química General e Inorgánica del primer semestre usando la codificación
## mencionada arriba
S1UC3 = UC(nombre = "Química General e Inorgánica", code = 'S1UC3',
            ucregulares = None, ucaprobadas = None,
            semestre = 1, créditos = 8)

## Creamos la UC Números Complejos y Ecuaciones Diferenciales del segundo
# semestre usando la codificación mencionada arriba
S2UC2 = UC(nombre = "Números Complejos y Ecuaciones Diferenciales", code = 'S2UC2',
            ucregulares = [S1UC1], ucaprobadas = None,
            semestre = 1, créditos = 8)

## Creamos la UC Óptiva y Radiaciones del tercer semestre
# semestre usando la codificación mencionada arriba
S3UC2 = UC(nombre = "Números Complejos y Ecuaciones Diferenciales", code = 'S3UC2',
            ucregulares = [S2UC2,S2UC3], ucaprobadas = [S1UC1,S1UC2],
            semestre = 1, créditos = 8)
## Podemos ver que la UC de Óptica necesita regulares las UC [S2UC2,S2UC3] y
## aprobadas las UC [S1UC1,S1UC2].


"""
*********************************************************
    Objeto plan de estudio IBIO año 2019
*********************************************************

Podríamos pensar en diseñar una clase llamada IBIO2019 que represente el plan de estudio
de Ingeniería Biomédica año 2018. Este plan de estudio podría tener, entre otras cosas,
todas las UCs (objetos) que lo conforman. Además podría tener un método, entre otros,
para acceder por ejemplo, a ver qué UCs conforman el semestre 1 de la carrera.
"""

ibio2019 = IBIO2019() #genero una instancia (objeto) del tipo IBIO2019
print(ibio2019.getUCs(semestre = 1)) #imprimo las UCs del semestre 1
## El programa podría imprimir un diccionario donde los keys podrían ser los códigos de las UC
## y los valores los nombres, por ejemplo,
## {'S1UC1': "Álgebra, Análisis y Geometría Analítica", 'S1UC2': Mecánica, Ondas y Calor,
##  'S1UC3': 'Química General e Inorgánica'... etc}
## O bien podría retornar una lista con objetos UCs, de la forma,
## [S1UC1, S1UC2, S1UC3, S1UC4, S1UC5, S1UC6, S1UC7]
## Sabemos que cada objeto UC posee métodos para obtener su nombre, etc.

## Podríamos pensar en que el objeto ibio2019 tenga un método para ver cuales
## son las previaturas de una UC en particular. Podríamos pensar que el
## método retorna dos listas, una con las UCs necesarias como regulares
## y otra con las aprobadas.

ibio2019.getPreviaturas(UC = "S3UC2")
## Podría retornar algo cómo,
## ([S2UC2,S2UC3], [S1UC1,S1UC2])
## Estas listas, idealmente, deberían contener objetos UC.
```