Instituto de Formación Técnica Superior Nro 18

CARRERA de TÉCNICO SUPERIOR en DESARROLLO DE SOFTWARE

Trabajo Práctico Final Integrador

Materia: Desarrollo de Sistemas Orientados a Objetos

Curso: 1er Año

Profesor: Lic. Eduardo Iberti

Ciclo Lectivo: 1er Cuatrimestre 2024

Fecha límite de entrega: lunes 01 de Julio de 2024

El trabajo se debe realizar en grupo de cinco alumnos.

Tema: Sistema de gestión de obras urbanas con manejo de POO, importación de datasets desde un archivo csv y persistencia de objetos con ORM Peewee en una base de datos SQLite.

Se requiere desarrollar un software en Python para gestionar las obras urbanas de la Ciudad de Buenos Aires, tomando como origen de datos un dataset público del gobierno de la ciudad y haciendo uso del modelo ORM de la librería ó módulo **"peewee"**. Para el manejo y operaciones con el dataset se debe utilizar la librería ó módulo **"pandas"**. Y para el manejo y operaciones con arrays pueden utilizar la librería ó módulo **"numpy"**.

Requerimientos:

1. En primer lugar, se debe **crear una carpeta para el proyecto solución del TP**, cuyo nombre debe contener el apellido de cada alumno integrante del equipo (cada apellido separado por guión).
2. Utilizar el **archivo csv "observatorio-de-obras-urbanas.csv"** que contiene los datos de las obras urbanas. Lo pueden descargar desde:

   https://data.buenosaires.gob.ar/dataset/ba-obras

   https://cdn.buenosaires.gob.ar/datosabiertos/datasets/secretaria-general-y-relaciones-internacionales/ba-obras/observatorio-de-obras-urbanas.csv.

   Ubicarlo en la carpeta del proyecto y analizar su estructura de datos **(ver estructura al pie del presente documento)**.
3. Crear el **módulo "modelo_orm.py"** que contenga la definición de las clases y atributos que considere necesarios, siguiendo el modelo **ORM de Peewee** para poder persistir los datos importados del dataset en una base de datos relacional de tipo SQLite llamada **"obras_urbanas.db"**, ubicada en la misma carpeta solución del proyecto. Aquí se debe incluir además la clase BaseModel heredando de peewee.Model.

4. Crear otro **módulo "gestionar_obras.py"** que contenga la definición de la **clase abstracta "GestionarObra"** y los siguientes métodos de clase:
    a. **extraer_datos()**, que debe incluir las sentencias necesarias para manipular el dataset a través de un objeto Dataframe del **módulo "pandas"**.
    b. **conectar_db()**, que debe incluir las sentencias necesarias para realizar la conexión a la base de datos "obras_urbanas.db".
    c. **mapear_orm()**, que debe incluir las sentencias necesarias para realizar la creación de la estructura de la base de datos (tablas y relaciones) utilizando el método de instancia "create_tables(list)" del **módulo "peewee"**.
    d. **limpiar_datos()**, que debe incluir las sentencias necesarias para realizar la "limpieza" de los datos nulos y no accesibles del Dataframe.
    e. **cargar_datos()**, que debe incluir las sentencias necesarias para persistir los datos de las obras (ya transformados y "limpios") que contiene el objeto Dataframe en la base de datos relacional SQLite. Para ello se debe utilizar el método de clase Model create() en cada una de las clase del modelo ORM definido.
    f. **nueva_obra()**, que debe incluir las sentencias necesarias para crear nuevas instancias de Obra. Se deben considerar los siguientes requisitos:
        - Todos los valores requeridos para la creación de estas nuevas instancias deben ser ingresados por teclado.
        - Para los valores correspondientes a registros de tablas relacionadas (foreign key), el valor ingresado debe buscarse en la tabla correspondiente mediante sentencia de búsqueda ORM, para obtener la instancia relacionada, si el valor ingresado no existe en la tabla, se le debe informar al usuario y solicitarle un nuevo ingreso por teclado.
        - Para persistir en la BD los datos de la nueva instancia de Obra debe usarse el método save() de Model del **módulo "peewee"**.
        - Este método debe retornar la nueva instancia de obra.
    g. **obtener_indicadores()**, que debe incluir las sentencias necesarias para obtener información de las obras existentes en la base de datos SQLite a través de sentencias ORM.
5. La **clase "Obra"**, que es una de las clases que debe formar parte del modelo ORM, debe incluir los siguientes métodos de instancia con el objetivo de definir las diferentes **etapas de avance de obra**:
    a. nuevo_proyecto().
    b. iniciar_contratacion().
    c. adjudicar_obra().
    d. iniciar_obra().
    e. actualizar_porcentaje_avance().
    f. incrementar_plazo().
    g. incrementar_mano_obra().
    h. finalizar_obra().
    i. rescindir_obra().
6. Se deberán crear nuevas instancias de Obra (dos instancias como mínimo) invocando al método de clase "GestionarObra.nueva_obra()".
7. Cada una de **las nuevas obras deben pasar por todas las etapas definidas**, salvo **incrementar_plazo()** e **incrementar_mano_obra()** que son opcionales. Para ello se debe

invocar a los métodos de instancia de la clase Obra, siguiendo el orden de la declaración de las etapas (desde **nuevo_proyecto()** hasta **finalizar_obra()** ó **rescindir_obra()**). Luego de cada cambio de estado del objeto Obra producto de una nueva etapa de avance de la obra, se deben persistir los nuevos valores usando el método save().

8. Para **iniciar un nuevo proyecto de obra se debe invocar al método nuevo_proyecto()**. Aquí la etapa inicial de las nuevas instancias de Obra debe tener el valor "Proyecto" (si este valor no existe en la tabla "etapas" de la BD, se deberá crear la instancia y luego insertar el nuevo registro). **Los valores de los atributos tipo_obra, area_responsable y barrio deben ser alguno de los existentes en la base de datos**.

9. A continuación, se debe **iniciar la licitación/contratación de la obra**, para ello se debe invocar al método **iniciar_contratacion()**, asignando el TipoContratacion (debe ser un valor existente en la BD) y el nro_contratacion.

10. Para **adjudicar la obra a una empresa, se debe invocar al método adjudicar_obra()** y asignarle la Empresa (debe ser una empresa existente en la BD) y el nro_expediente.

11. Para indicar el **inicio de la obra, se debe invocar al método iniciar_obra()**, y asignarle valores a los siguientes atributos: destacada, fecha_inicio, fecha_fin_inicial, fuente_financiamiento (debe ser un valor existente en la BD) y mano_obra.

12. Para registrar **avances de la obra, se debe invocar al método actualizar_porcentaje_avance()** y actualizar el valor del atributo porcentaje_avance.

13. Para **incrementar el plazo de la obra, se debe invocar al método incrementar_plazo()** y actualizar el valor del atributo plazo_meses. **(Esta acción es opcional, pero el método debe estar definido).**

14. Para **incrementar la cantidad de mano de obra, se debe invocar al método incrementar_mano_obra()** y actualizar el valor del atributo mano_obra. **(Esta acción es opcional, pero el método debe estar definido)**.

15. Para **indicar la finalización de una obra, se debe invocar al método finalizar_obra()** y actualizar el valor del atributo etapa a "Finalizada" y del atributo porcentaje_avance a "100".

16. Para **indicar la rescisión de una obra, se debe invocar al método rescindir_obra()** y actualizar el valor del atributo etapa a "Rescindida".

17. Para finalizar la ejecución del programa, se debe **invocar al método de clase GestionarObra.obtener_indicadores()** para obtener y mostrar por consola la siguiente información:

    a. Listado de todas las áreas responsables.
    b. Listado de todos los tipos de obra.
    c. Cantidad de obras que se encuentran en cada etapa.
    d. Cantidad de obras y monto total de inversión por tipo de obra.
    e. Listado de todos los barrios pertenecientes a las comunas 1, 2 y 3.
    f. Cantidad de obras finalizadas y su y monto total de inversión en la comuna 1.
    g. Cantidad de obras finalizadas en un plazo menor o igual a 24 meses.

h. Porcentaje total de obras finalizadas.
i. Cantidad total de mano de obra empleada.
j. Monto total de inversión.

Aclaraciones:

1. En la **clase abstracta "GestionarObra", todos sus métodos deben ser métodos de clase y sus atributos** (en caso que considere que deba existir alguno) **también serán atributos de clase.**
2. Incluir código Python para **manejar posibles excepciones** donde considere conveniente para atrapar aquellas que puedan llegar a generarse.

Qué es la librería o módulo "Pandas"?

Esta biblioteca de software de código abierto está diseñada específicamente para la manipulación y el análisis de datos en el lenguaje Python. Es potente, flexible y fácil de usar.

Pandas se puede utilizar en el lenguaje Python para cargar, alinear, manipular o incluso fusionar datos.

Facilita el procesamiento de datos estructurados en forma de tablas, matrices o series temporales. También es compatible con otras bibliotecas de Python.

Pandas trabaja sobre "DataFrames" (tablas de datos bidimensionales), donde cada columna contiene los valores de una variable y cada fila contiene un conjunto de valores de cada columna. Los datos almacenados en un DataFrame pueden ser números o caracteres.

Los Data Scientists y los programadores familiarizados con el lenguaje de programación R para cálculo estadístico utilizan DataFrames para almacenar datos en una cuadrícula muy sencilla de revisar. Por eso Pandas se utiliza mucho para Machine Learning.

Esta herramienta permite importar y exportar datos en distintos formatos, como CSV o JSON.

https://datascientest.com/es/pandas-python

https://www.w3schools.com/python/pandas/default.asp

https://pypi.org/project/pandas/

Qué es la librería o módulo "NumPy"?

NumPy es una librería de Python especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos.

Incorpora una nueva clase de objetos llamados **arrays** que permite representar colecciones de datos de un mismo tipo en varias dimensiones, y funciones muy eficientes para su manipulación.

La ventaja de Numpy frente a las listas predefinidas en Python es que el procesamiento de los arrays se realiza mucho más rápido (hasta 50 veces más) que las listas, lo cual la hace ideal para el procesamiento de vectores y matrices de grandes dimensiones.

https://aprendeconalf.es/docencia/python/manual/numpy/

https://facundoq.github.io/courses/images/res/03_numpy.html

https://joserzapata.github.io/courses/python-ciencia-datos/numpy/

https://geekflare.com/es/numpy-arrays/

Estructura del dataset "Obras Públicas"

Información de las obras públicas realizadas en la Ciudad. Información sobre empresa constructora, CUIT, fecha de inicio y fin, monto, descripción, ubicación y ministerio responsable de las obras públicas realizadas por el Gobierno de la Ciudad.

Nombre	Tipo	Descripción
id	string	id
entorno	string	entorno
nombre	string	nombre
etapa	string	etapa
tipo	string	tipo
area_responsable	string	area_responsable
descripcion	string	descripcion
monto_contrato	string	monto_contrato
comuna	string	comuna
barrio	string	barrio
direccion	string	direccion
lat	string	lat
lng	string	lng

fecha_inicio	string	fecha_inicio
fecha_fin_inicial	string	fecha_fin_inicial
plazo_meses	string	plazo_meses
porcentaje_avance	string	porcentaje_avance
imagen_1	string	imagen_1
imagen_2	string	imagen_2
imagen_3	string	imagen_3
imagen_4	string	imagen_4
licitacion_oferta_empresa	string	licitacion_oferta_empresa
licitacion_anio	string	licitacion_anio
contratacion_tipo	string	contratacion_tipo
nro_contratacion	string	nro_contratacion
cuit_contratista	string	cuit_contratista
beneficiarios	string	beneficiarios
mano_obra	string	mano_obra
compromiso	string	compromiso
destacada	string	destacada
ba_elige	string	ba_elige
link_interno	string	link_interno
pliego_descarga	string	pliego_descarga
expediente-numero	string	expediente-numero
estudio_ambiental_descarga	string	estudio_ambiental_descarga
financiamiento	string	financiamiento