Facultad de Ciencias de la Administración
UNER

TRABAJO FINAL INTEGRADOR

PROGRAMACIÓN II - 2025 – 2do cuatrimestre

TECNICATURA UNIVERSITARIA EN DESARROLLO WEB

Objetivos

- Aplicar y reforzar los conceptos adquiridos durante toda la cursada.

Condiciones de Entrega

- Deberá realizarse una entrega grupal, con grupos de no más de 3 (tres) alumnos.
- Se va a brindar como base para el trabajo una carpeta comprimida que contiene un sistema incompleto. La entrega deberá consistir de una carpeta comprimida que contenga los archivos de la base proveída más los archivos y modificaciones solicitados en las consignas de este documento. Se deberá indicar con claridad la sección del código que se corresponde con cada uno de los ejercicios del documento. La carpeta debe ser subida en la sección correspondiente del Campus.
- Se deberá respetar la fecha de entrega, la misma será informada en el campus.
- El Trabajo Práctico será calificado como **Aprobado** o **Desaprobado**.
- Las soluciones del grupo deberán ser de autoría propia. De encontrarse soluciones idénticas entre diferentes grupos, dichos trabajos prácticos serán calificados como **Desaprobado**, lo cual será comunicado en la devolución.
- Las entregas individuales serán calificadas como **Desaprobado**.

EJERCICIOS:

***Nota**: asumir que todos los parámetros recibidos en los métodos correspondientes a los ejercicios del trabajo son del tipo de datos correcto, con excepción de los casos donde se especifique la necesidad de agregar validaciones.*

***Nota:** para iniciar el sistema, se debe ejecutar:*

```
pip install flask
```

```
python -m flask run
```

Se provee como base un servidor http desarrollado con Python y Flask, que utiliza como fuente de datos precargados un archivo con formato JSON. El sistema, una vez esté funcional, debería ser capaz de recuperar los datos del archivo JSON, crear objetos a partir de los mismos, y retornar una representación de dichos objetos como cadena de texto formateada.

Se va trabajar sobre un modelo de concesionarias, incluyendo su nombre, clientes, vehículos, y sucursales, de las cuales se registran las ventas realizadas. Se provee el código de las clases correspondientes a los siguientes diagramas

En el archivo concesionaria.py:

Concesionaria
<<Atributos de instancia>> numeroId: int nombre: string clientes: Cliente[] sucursales: Sucursal[] vehiculos: Vehiculo[]
<<Constructores>> Concesionaria(numeroId: int, nombre: string) <<Comandos>> establecerNumeroId(numeroId: int) establecerNombre(nombre: string) agregarCliente(cliente: Cliente) removerCliente(cliente: Cliente) agregarSucursal(sucursal: Sucursal) removerSucursal(sucursal: Sucursal) agregarVehiculo(vehiculo: Vehiculo) removerVehiculo(vehiculo: Vehiculo) <<Consultas>> obtenerNumeroId(): int obtenerNombre(): string obtenerClientes(): Cliente[] obtenerSucursales(): Sucursal[] obtenerVehiculos(): Vehiculo[]

En el archivo vehiculo.py:

Vehiculo
<<Atributos de clase>> ESTADO_DISPONIBLE_ID = 1 ESTADO_VENDIDO_ID = 2 ESTADO_BAJA_ID = 3 <<Atributos de instancia>> numeroId:int marca:string modelo:string anio:int sucursalId: int estadoId: int
<<Constructores>> Vehiculo(numeroId:int, marca:string, modelo:string, anio:int, sucursalId: int, estadoId: int) <<Comandos>> establecerNumeroId(numeroId:int) establecerMarca(marca:string) establecerModelo(modelo: string) establecerAnio(anio: int) establecerSucursalId(sucursalId: int) establecerEstadoId(estadoId: int) <<Consultas>> obtenerNumeroId(): int obtenerMarca(): string obtenerModelo(): string obtenerAnio(): int obtenerSucursalId(): int obtenerEstadoId(): int

En el archivo moto.py:

Moto (Vehiculo)
<<Atributos de instancia>> cilindrada:int
<<Constructores>> Moto(numeroId:int, marca:string, modelo:string, anio:string, sucursalId: int, estadoId: int, cilindrada: int) <<Comandos>> establecerCilindrada(cilindrada: int) <<Consultas>> obtenerCilindrada(): int

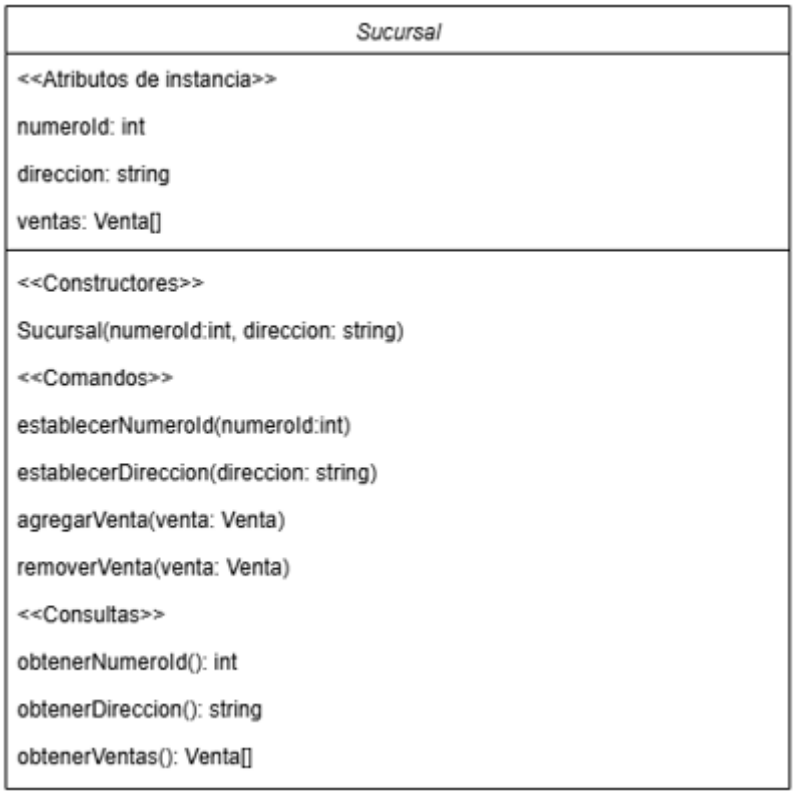
En el archivo servicio_concesionarias.py:

ServicioConcesionarias
<<Consultas>> obtenerPorId(concesionariaId): Concesionaria

Nota: El método "obtenerPorId" se provee ya implementado, y se encarga de recuperar los datos del archivo JSON correspondientes a la concesionaria indicada.

Y se exige que se completen los siguientes ítems:

1) En el archivo sucursal.py, generar la clase correspondiente al siguiente diagrama:



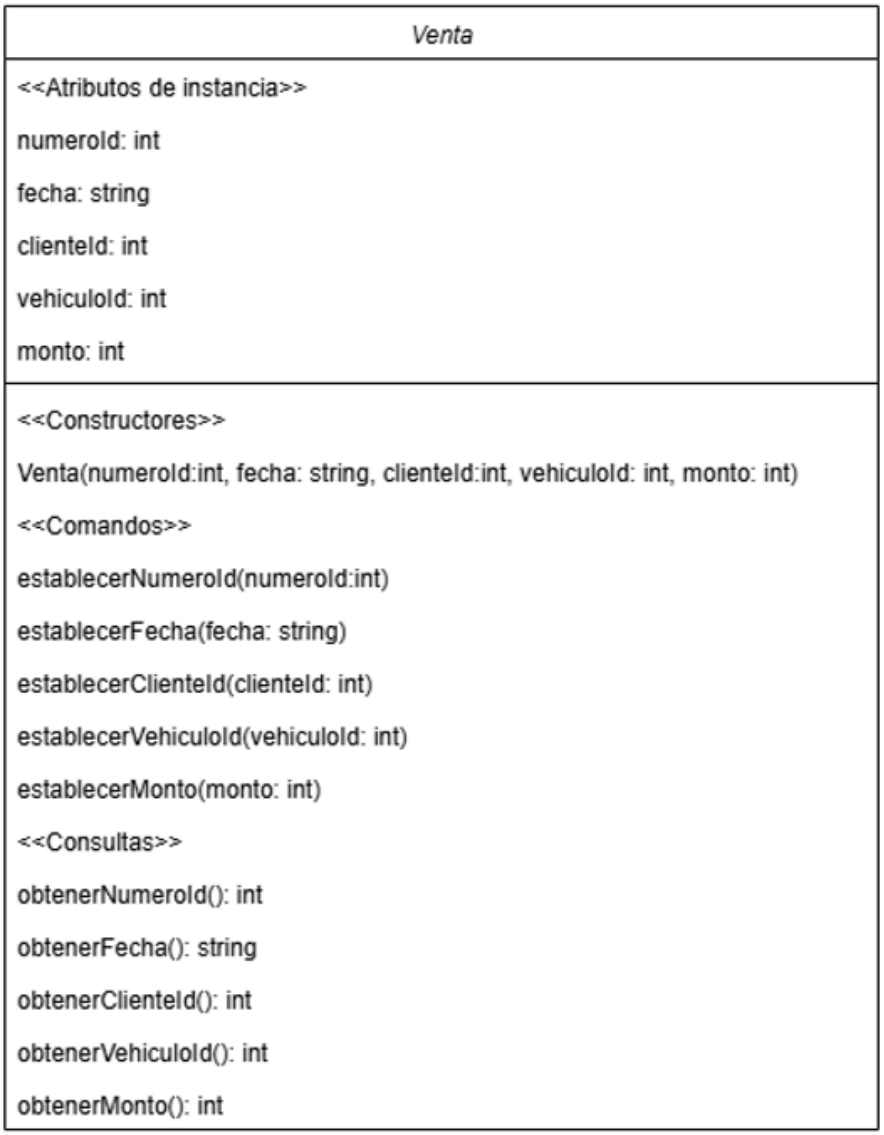
2) En el archivo cliente.py, generar la clase correspondiente al siguiente diagrama:

Cliente
<<Atributos de instancia>> numeroId:int nombres: string apellidos: string email: string
<<Constructores>> Cliente(numeroId:int, nombres: string, apellidos: string, mail: string) <<Comandos>> establecerNumeroId(numeroId :int) establecerNombres(nombres: string) establecerApellidos(apellidos:string) establecerEmail(email: string) <<Consultas>> obtenerNumeroId():int obtenerNombres(): string obtenerApellidos(): string obtenerEmail(): string

3) En el archivo auto.py, generar la clase correspondiente al siguiente diagrama:

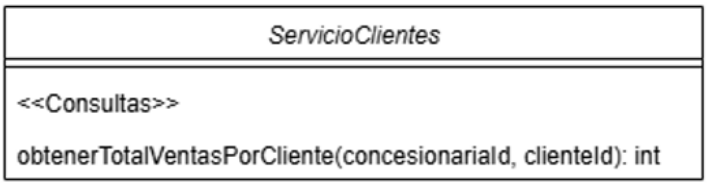
Auto (Vehiculo)
<<Atributos de instancia>> airbags:int frenosAbs: boolean caballosFuerza: int
<<Constructores>> Auto(numeroId:int, marca:string, modelo:string, anio:string, sucursalId: int, estadoId: int, airbags: int, frenosAbs:boolean, caballosFuerza: int) <<Comandos>> establecerAirbags(aribags: int) establecerFrenosAbs(frenosAbs:boolean) establecerCaballosFuerza(caballosFuerza: int) <<Consultas>> obteneAirbags(): int obtenerFrenosAbs(): boolean obtenerCaballosFuerza(): int

4) En el archivo venta.py, generar la clase correspondiente al siguiente diagrama:



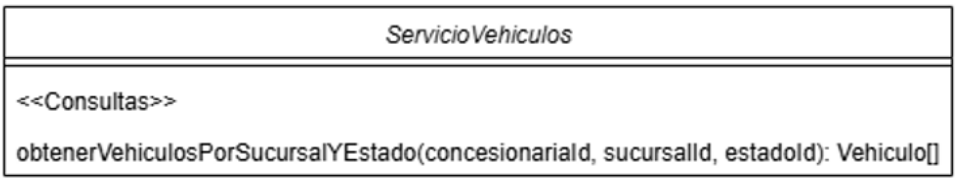
5) Actualizar tanto las clases proveídas (con excepción de ServicioConcesionarias) como también las clases creadas en los puntos anteriores para incluir el método __eq__, comparando en todos los casos por "numero_id" (con excepción de las subclases de "Vehiculo", que deben heredar __eq__ de la superclase), y el método __str__, mostrando en todos los casos la información correspondiente a todas las propiedades de cada clase (con excepción de la clase "Vehiculo", que declara el método como abstracto para ser implementado en las subclases).

6) En el archivo servicio_clientes.py, implementar el método "obtenerTotalVentasPorCliente" de la clase correspondiente al siguiente diagrama:



La clase y el método se proveen definidos, pero es necesario implementar la lógica interna del método, que se provee con "pass". El mismo debe retornar un número entero calculado como la suma de los montos de todas las ventas realizadas al cliente indicado en la concesionaria indicada. El mismo debe retornar 0 en caso de que la concesionaria no exista, el cliente no exista, o no existan ventas al cliente indicado en la concesionaria indicada. Se debe utilizar el método "obtenerPorId" de la clase "ServicioConcesionarias" para recuperar todos los datos disponibles de la concesionaria pertinente, y trabajar sobre los mismos para obtener el resultado solicitado.

7) En el archivo servicio_vehiculos.py, implementar el método "obtenerVehiculosPorSucursalYEstado" de la clase correspondiente al siguiente diagrama:



La clase y el método se proveen definidos, pero es necesario implementar la lógica interna del método, que se provee con "pass". El mismo debe retornar una lista de todos los vehículos que se encuentren en el estado indicado, en la sucursal indicada de la concesionaria indicada. El mismo debe retornar una lista vacía en caso de que la concesionaria no exista, la sucursal no exista, o no existan vehículos con el estado indicado en la sucursal indicada de la concesionaria indicada. Se debe utilizar el método "obtenerPorId" de la clase "ServicioConcesionarias" para recuperar todos los datos disponibles de la

concesionaria pertinente, y trabajar sobre los mismos para obtener el resultado solicitado.

Enlaces para verificación de correctitud:

Para auxiliar el proceso de verificación de correctitud del funcionamiento del sistema, pueden hacer click a los siguientes enlaces dentro de un navegador:

http://localhost:5000/concesionarias/1/

El mismo retorna el resultado de ejecutar str() sobre la concesionaria con numero_id igual a 1 (la única registrada en el archivo JSON proveído). Se deberían visualizar todos los datos registrados de la concesionaria, incluyendo vehículos, clientes, y sucursales, cada uno con todos sus datos correspondientes, incluidas las ventas de las sucursales.

http://localhost:5000/concesionarias/1/clientes/2/total-ventas/

El mismo retorna como resultado el monto total de todas las ventas al cliente con numero_id igual a 2 en la concesionaria con numero_id igual a 1 (la única registrada en el archivo JSON proveído), en este caso 3000000.

http://localhost:5000/concesionarias/1/sucursales/2/vehiculos?estado_id=1

El mismo retorna como resultado información de todos los vehículos (cada vehículo siendo representado con un string generado con str()) cuyo atributo estado_id tenga valor 1 que pertenezcan a la sucursal con numero_id igual a 2 de la concesionaria con numero_id igual a 1 (la única registrada en el archivo JSON proveído).