Facultad de Ciencias
UNER de la Administración

TRABAJO INTEGRADOR FINAL

PROGRAMACIÓN 2 - 2024 – 2do cuatrimestre

TECNICATURA UNIVERSITARIA EN DESARROLLO WEB

EL TRABAJO INTEGRADOR FINAL TIENE POR OBJETIVO QUE EL ALUMNO

- Aplique y refuerce los conceptos adquiridos durante toda la cursada.
- Sea capaz de programar un aplicativo mínimo aplicando dichos conocimientos, interpretando y traduciendo correctamente los diagramas de Clases en código Python respetando las relaciones y responsabilidades entre las Clases y los Objetos creados a partir de ellas.

CONDICIONES DE ENTREGA

- El Trabajo Práctico deberá ser:
    - Realizado en forma **grupal, en equipos de entre 3 y 5 alumnos**.
    - Cargado en la sección del Campus Virtual correspondiente, en un archivo 7z, ZIP o RAR (o cualquier otro tipo de comprimido) con las soluciones a cada ejercicio. Cada solución debe estar contenida en un archivo .py distinto.
    - Deberá indicarse el apellido y nombre de los integrantes del grupo. Todos los integrantes del grupo deben realizar la entrega en el campus y deberá agregarse al comprimido con las soluciones un archivo *integrantes.txt* con la información de los participantes.
    - Entregado antes de la fecha límite informada en el campus.
- El Trabajo Práctico será calificado como Aprobado o Desaprobado.
- Las soluciones del alumno/grupo deben ser de autoría propia. De encontrarse soluciones idénticas entre diferentes grupos, dichos trabajos prácticos serán clasificados como **DESAPROBADO**, lo cual será comunicado en la devolución.

VINOTECA VIRTUAL

Se presenta un repositorio de código que los alumnos deben terminar de codificar. En el mundo de los aplicativos web y móviles, es común encontrar diseños donde los datos están contenidos en algún lugar, al cual los aplicativos acceden consumiendo distintos tipos de servicios web (rest, SOAP, websockets, etc). El código ofrecido refiere a un aplicativo que expone servicios web para realizar consultas sobre la base de datos de una vinoteca virtual contenida en un archivo JSON.

Los servicios que este aplicativo ofrece son los siguientes:

1. http://localhost:5000/api/bodegas/<id>
   Se obtiene la información de una bodega, los vinos y cepas que esta ofrece.
2. http://localhost:5000/api/bodegas
   Se consulta el listado completo de bodegas, las cepas y la cantidad de vinos que estas ofrecen.
3. http://localhost:5000/api/cepas/<id>
   Se obtiene la información de una cepa y los vinos y bodegas que la ofrecen en sus botellas.
4. http://localhost:5000/api/cepas
   Se consulta el listado completo de cepas y los vinos y bodegas que las ofrecen en sus botellas.
5. http://localhost:5000/api/vinos/<id>
   Se obtiene la información de un vino, la bodega que lo produce, las cepas en las que se lo ofrece, y las partidas con las que se cuenta para él.
6. http://localhost:5000/api/vinos
   Se consulta el listado completo de vinos, las bodegas que los producen, las cepas en las que se los ofrece, y las partidas con las que se cuenta para ellos.

Los servicios 2., 4. y 6. pueden recibir los parámetros opcionales **orden** y **reverso** (que únicamente puede tomar los valores *si* o *no*), los cuales indiquen el campo por el cual las listas deben ser ordenadas y si el orden debe ser el regular o inverso (ver ejemplos en la sección **Ejecución y Prueba**). Además, el servicio 6 puede tomar el parámetro adicional **anios** el cual indique que se recuperen los vinos que incluyan la partida indicada.

A lo largo de este trabajo los estudiantes deberán codificar los modelos de datos como así también parte de la lógica interna de las consultas expuestas como servicios web. Esto es, en principio, traducir los diagramas de clases a código Python y escribir la lógica que permita recuperar los datos del archivo JSON para ser convertidos a objetos de dichas clases y finalmente ser mostrados al usuario.

El aplicativo está soportado por el framework Flask, por lo que deberá previamente tenerlo instalado para poder correrlo. Para instalarlo corra el siguiente comando por consola:

pip install flask

pip install Flask-RESTful

Para aquellos que corran una versión más avanzada de Python, puede que deban utilizar los siguientes comandos en lugar de los anteriores:

pip3 install flask

pip3 install Flask-RESTFul

De requerirse más detalles sobre la instalación, uso u otras características del framework en cuestión, por favor referirse a la documentación oficial alojada en el siguiente link:

https://flask.palletsprojects.com/en/2.2.x/

Modelado de Datos

El modelo de datos presentado para esta solución está comprendido por la definición de 3 entidades. Estas son las Bodegas, las Cepas y los Vinos. Todas estas entidades poseen dos atributos de instancia comunes a todas ellas, por lo que esta estructura estará comprendida en la clase abstracta **EntidadVineria**.

1. Implementar los siguientes diagramas de clases en los archivos indicados del paquete modelos:

EntidadVineria
<<Atributos de clase>> <<Atributos de instancia>> id: string nombre: string
<<Constructores>> EntidadVineria(id, nombre: string) <<Comandos>> establecerNombre(nombre: string) <<Consultas>> obtenerId(): string obtenerNombre(): string

   a. Utilizar el archivo *entidadvineria.py*
   b. Se debe sobreescribir la consulta __***eq***__ para que compare dos objetos de la clase por el atributo de instancia id.
   c. EntidadVineria debe ser una clase abstracta, es decir, no debe poder instanciarse objetos de dicha clase directamente.

Bodega (EntidadVineria)
<<Atributos de clase>> <<Atributos de instancia>>
<<Constructores>> Bodega(id, nombre: string) <<Comandos>> <<Consultas>> obtenerVinos(): Vino[] obtenerCepas(): Cepa[] convertirAJSON(): dict convertirAJSONFull(): dict

a. Utilizar el archivo *bodega.py*
b. La consulta ***obtenerVinos*** debe hacer uso del servicio ***obtenerVinos*** de la clase **Vinoteca** para recuperar todos los vinos contenidos en el archivo json. Sobre dicha lista se debe iterar hasta encontrar los vinos que pertenecen a la bodega en cuestión.
c. La consulta ***obtenerCepas*** debe seguir la misma impronta que el punto anterior para intentar encontrar aquellos vinos que pertenecen a la bodega, recuperando únicamente las cepas en los que se ofrecen estos.

Cepa (EntidadVineria)
<<Atributos de clase>> <<Atributos de instancia>>
<<Constructores>> Cepa(id, nombre: string) <<Comandos>> <<Consultas>> obtenerVinos(): Vino[] convertirAJSON(): dict convertirAJSONFull(): dict

a. Utilizar el archivo *cepa.py*
b. La consulta ***obtenerVinos*** debe hacer uso del servicio ***obtenerVinos*** de la clase **Vinoteca** para recuperar todos los vinos contenidos en el archivo json. Sobre dicha lista se debe iterar hasta encontrar los vinos que se ofrecen en la cepa en cuestión.

Vino (EntidadVineria)
<<Atributos de clase>> <<Atributos de instancia>> bodega: string cepas: string[] partidas: int[]
<<Constructores>> Vino(id, nombre, bodega, cepas: string[], cepas: int[]) <<Comandos>> establecerNombre(nombre: string) establecerBodega(bodega: string) establecerCepas(cepas: string[]) establecerPartidas(partidas: int[]) <<Consultas>> obtenerId(): string obtenerNombre: string obtenerBodega: Bodega obtenerCepas: Cepa[] obtenerPartidas: int[] convertirAJSON(): dict convertirAJSONFull(): dict

a. Utilizar el archivo *vino.py*
b. La consulta ***obtenerBodega*** debe hacer uso del servicio ***buscarBodega*** de la clase **Vinoteca** para recuperar el objeto de tipo **Bodega** asociado al vino.
c. La consulta ***obtenerCepas*** puede hacer uso de los servicios ***buscarCepa*** u ***obtenerCepas*** de la clase **Vinoteca** para recuperar los objetos de tipo Cepa para las cepas en las que se ofrece el vino en cuestión.

Motor de Consultas

Vinoteca es una clase que no posee atributos de instancia. Esto es así porque no sigue el propósito de crear objetos de tipo Vinoteca a partir de ella, sino como entidad para centralizar las consultas a realizarse sobre la base datos. Entonces, las entidades que modelan los objetos del mundo (Bodega, Cepa, Vino) deberán hacer uso de las consultas de la clase Vinoteca, enviando mensajes directamente a la Clase en lugar de un objeto creado a partir de ella.

2. Dada la clase **Vinoteca**, contenida en *vinoteca.py*, y que responde al siguiente diagrama:

Vinoteca
<<Atributos de clase>> archivoDeDatos = "vinoteca.json" bodegas = Bodega[] cepas = Cepa[] vinos = Vino[] <<Atributos de instancia>>
<<Constructores>> <<Comandos>> inicializar() <<Consultas>: obtenerBodegas(orden, reverso: string): Bodega[] obtenerCepas(orden, reverso: string): Cepas[] obtenerVinos(anio: int, orden, reverso: string): Vino[] buscarBodega(id: string): Bodega buscarCepa(id: string): Cepa buscarVino(id: string): Vino

   a. Utilizar las colecciones *Vinoteca.bodegas*, *Vinoteca.cepas*, *Vinoteca.vinos*para codificar las consultas según corresponda.
   b. Para las consultas ***obtenerBodegas***, ***obtenerCepas*** y ***obtenerVinos***, si se recibe:

i. Los parámetros opcionales **orden <> None** y **reverso = True**, devolver una copia de la colección correspondiente, ordenando los objetos de la misma por el atributo indicado en orden inverso.
ii. Ninguno de los parámetros opcionales, devolver una referencia a la colección correspondiente sin alterar su orden.

c. Para la consulta ***obtenerVinos***, si se recibe el parámetro **anio <> None**, antes de ordenar la colección a retornar, esta debe filtrarse manteniendo únicamente aquellos vinos cuyas partidas contengan el año en cuestión.

d. Para las consultas ***buscarBodega***, ***buscarCepa***, ***buscarVino***:
i. Utilizar el parámetro formal **id** para navegar la colección y encontrar el elemento que coincida con dicho identificador.
ii. Si no se encontrase coincidencia, retornar el valor **None**.

e. Codificar la lógica del método interno ***__parsearArchivoDeDatos*** de manera que:
i. Abra el archivo de datos.
ii. Cargue la información en una estructura de tipo diccionario.
iii. Cierre el archivo.
iv. Retorne una referencia al diccionario anteriormente creado.

f. Codificar la lógica del método interno ***__convertirJsonAListas(listas: dict[])*** de la siguiente manera:
i. Se espera se completen las 3 colecciones:
1. *Vinoteca.bodegas*
2. *Vinoteca.cepas*
3. *Vinoteca.vinos*

Ejecución y Prueba

Al finalizar la codificación del trabajo, el alumno podrá verificar que su solución es correcta y cumple con los requisitos para aprobar este trabajo si es que al correr el aplicativo y al visitarse las siguientes URL a través de un navegador web, se obtienen los resultados que se muestra a continuación:

URL: http://127.0.0.1:5000/api/bodegas/a0900e61-0f72-67ae-7e9d-4218da29b7d8

Resultado Esperado:


```
{"id":"a0900e61-0f72-67ae-7e9d-4218da29b7d8","nombre":"Casa La Primavera Bodegas y Viñedos","cepas":["Chardonnay","Malbec","Cabernet Suavignon","Merlot"],"vinos":["Profugo","Oveja Black","Sin Palabra"]}
```


URL: http://127.0.0.1:5000/api/cepas/33ccaa9d-4710-9942-002d-1b5cb9912e5d

Resultado Esperado:


```
{"id":"33ccaa9d-4710-9942-002d-1b5cb9912e5d","nombre":"Chardonnay","vinos":["Profugo (Casa La Primavera Bodegas y Viñedos)","Oveja Black (Casa La Primavera Bodegas y Viñedos)","Sin Palabra (Casa La Primavera Bodegas y Viñedos)","Sottano (Bodega Sottano)"]}
```


URL: http://127.0.0.1:5000/api/vinos/4823ad54-0a3a-38b8-adf6-795512994a4f

Resultado Esperado:


```
{"id":"4823ad54-0a3a-38b8-adf6-795512994a4f","nombre":"Abducido","bodega":"La Iride","cepas":["Cabernet Suavignon","Malbec"],"partidas":[2024,2023,2022]}
```


URL: http://127.0.0.1:5000/api/vinos?**anio=2020**&orden=**nombre**&**reverso=no**

Resultado Esperado:


```
[{"id":"ea3d6c45-e747-2e86-ddf8-0746cc13f21c","nombre":"Familia Gascon","bodega":"Escorihuela Gascon","cepas":["Cabernet Suavignon","Malbec","Merlot"],"partidas":[2020,2021]},{"id":"205db14f-c0c0-a286-0a5e-0daa7a3f37f0","nombre":"Sin Palabra","bodega":"Casa La Primavera Bodegas y Viñedos","cepas":["Chardonnay","Cabernet Suavignon","Malbec","Merlot"],"partidas":[2022,2021,2020]}]
```

URL: http://127.0.0.1:5000/api/vinos?**anio=2020**&orden=**nombre**&**reverso=si**

Resultado Esperado:


```
[{"id":"205db14f-c0c0-a286-0a5e-0daa7a3f37f0","nombre":"Sin
Palabra","bodega":"Casa La Primavera Bodegas y
Viñedos","cepas":["Chardonnay","Cabernet
Suavignon","Malbec","Merlot"],"partidas":[2022,2021,2020]},{"id":"ea3d
6c45-e747-2e86-ddf8-0746cc13f21c","nombre":"Familia
Gascon","bodega":"Escorihuela Gascon","cepas":["Cabernet
Suavignon","Malbec","Merlot"],"partidas":[2020,2021]}]
```