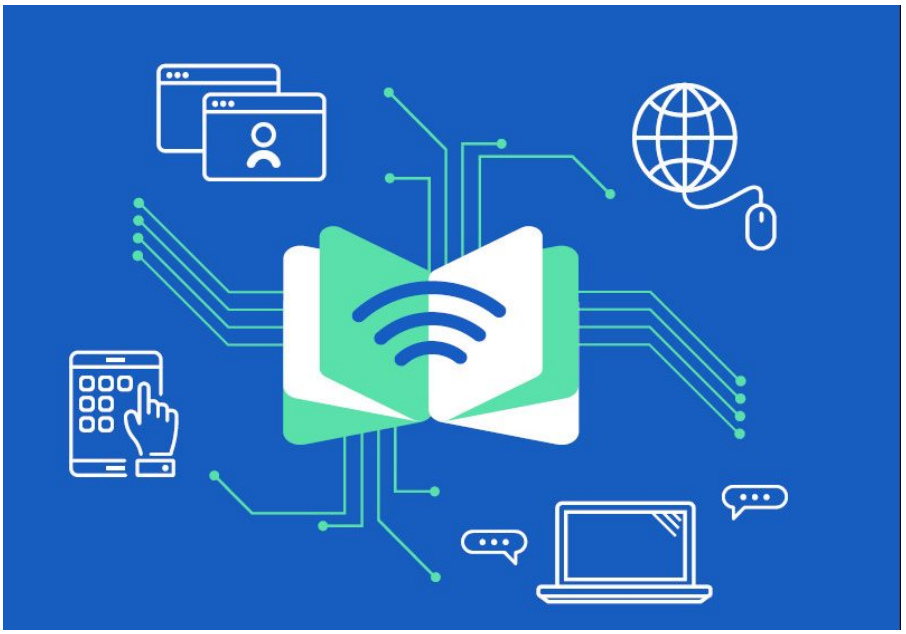
Algoritmos y Programación II
Facultad de ingeniería

TAREA INTEGRADORA 2



Usted y su compañero son contratados para desarrollar un novedoso **sistema de información** geográfica. Es un programa que en esta versión almacenará información de ***ciudades y países***.

Los países contarán con los siguientes atributos:
Id. De tipo String. Debe ser generado mediante UUID
Name**.** De tipo String. Es el parámetro con el nombre de la ciudad
Population. De tipo Double. Almacena la población actual de la ciudad en millones de personas
CountryCode. De tipo String. Representa el indicativo del país

Las ciudades contarán con los siguientes atributos:
Id. De tipo String. Debe ser generado mediante UUID
Name**.** De tipo String. Es el parámetro con el nombre de la ciudad
CountryID**.** De tipo String. Contendrá el identificador del país
Population. De tipo Integer. Almacena la población actual de la ciudad

El programa funcionará como un programa por consola. En el programa el usuario siempre tendrá a disposición este menú:

```
1. Insertar comando
2. Importar datos desde archivo .SQL
2. Salir
```

Al usar insertar comando, el usuario podrá agregar, buscar y filtrar, ordenar y eliminar datos.

1. AGREGAR DATOS

Mediante el comando INSERT INTO, el usuario podrá agregar un dato a la base de datos.

```
INSERT INTO countries(id, name, population, countryCode) VALUES
('6ec3e8ec-3dd0-11ed-b878-0242ac120002', 'Colombia', 50.2, '+57')
```

Ya con un país, podemos agregar la ciudad:

```
INSERT INTO cities(id, name, countryID, population) VALUES
('e4aa04f6-3dd0-11ed-b878-0242ac120002', 'Cali', '6ec3e8ec-3dd0-11ed-b878-0242ac120002', 2.2)
```

Al agregar el datos de Cali, note que el *countryID* corresponde al ID de Colombia.
Si el usuario intenta agregar una ciudad y usa un ID de un país inexistente en la base de datos, el programa debe mostrarlo como una ***excepción.***
Si uno de los datos de entrada no está con el formato correcto, también debería lanzar una excepción. Note que cuando el dato es String va entre comillas simples y si el dato es numérico, va sin comillas.

Por último también lanza una excepción si el comando escrito no es correcto. Analice la estructura del comando

2. BÚSQUEDA Y FILTRADO
Mediante el comando SELECT, el usuario podrá buscar algo en la base de datos. Por ejemplo:
SELECT * FROM cities WHERE name = 'Colombia'
Permitirá al usuario buscar en ***cities*** a un país llamado Colombia. El programa lo buscará y lo mostrará por consola si existe.

```
SELECT * FROM countries WHERE population > 100
```
Permitirá buscar todos los países con una población superior a 100 millones de personas

```
SELECT * FROM countries WHERE population < 30
```
Permitirá buscar todos los países con una población inferior a 30 millones de personas

```
SELECT * FROM countries
```
Si no tiene la palabra WHERE, mostrará al usuario todos los países.

Sólo maneje las comparaciones igual (=), mayor (>) y menor(<). Tenga en cuenta que cuando está trabajando con Strings, deben estar entre comillas simples para el comando y cuando maneje número, estos van sin comillas.

3. ORDENAMIENTO
A los comandos que inician con SELECT, también se les podrá combinar con ordenamientos, por ejemplo

```
SELECT * FROM countries WHERE population > 100 ORDER BY name
```
Permite listar a todos los países con población superior a 100 millones de personas y las muestra ordenadas por nombre.

```
SELECT * FROM cities WHERE name = 'Guadalajara' ORDER BY population
```
Permite listar a todas las ciudades llamadas Guadalajara y permite ordenarlas por población

Entonces los resultados que ofrezca el comando SELECT serán ordenados por el parámetro seguido al comando ORDER BY.

Si el resultado de la búsqueda sólo da un resultado, el ORDER BY será ignorado.

4. ELIMINACIÓN
El comando DELETE permite eliminar un registro. Por ejemplo:

DELETE FROM cities WHERE country = 'Colombia'
Elimina todos los países con nombre Colombia

DELETE FROM cities WHERE population > 50
Elimina todos los países con una población mayor a 50 millones de personas.

5. ALMACENAMIENTO E IMPORTACIÓN DE DATOS

Toda la información que se ingresa a la base de datos queda almacenada en formato JSON, de modo que el programa pueda recuperarla al iniciar el programa. Entonces el programa siempre debería estar actualizado respecto al último cambio en la base de datos.

El usuario también puede usar la opción 2 para importar datos desde un archivo .SQL. El programa debe poder importar datos desde cualquier archivo con extensión .sql que tenga el formato correcto. Un archivo .SQL es un archivo de texto que contiene los comandos INSERT del punto 1. Al leerlos, el programa ejecutará de forma secuencial cada línea. Un ejemplo de archivo .sql es el que se muestra a continuación:

```
INSERT INTO countries(id, name, population, countryCode) VALUES ('6ec3e8ec-3dd0-11ed-b878-0242ac120002', 'Colombia', 50.2)
INSERT INTO countries(id, name, population, countryCode) VALUES ('71615790-3dd2-11ed-b878-0242ac120002', 'USA', 329.5)
INSERT INTO countries(id, name, population, countryCode) VALUES ('83b3e642-3dd2-11ed-b878-0242ac120002', 'México', 128.9)
INSERT INTO cities(id, name, countryID, population) VALUES ('e4aa04f6-3dd0-11ed-b878-0242ac120002', 'Cali', '6ec3e8ec-3dd0-11ed-b878-0242ac120002', 2.2)
INSERT INTO cities(id, name, countryID, population) VALUES ('8a8f23a0-3dd2-11ed-b878-0242ac120002', 'Washington DC', '71615790-3dd2-11ed-b878-0242ac120002',701.9)
INSERT INTO cities(id, name, countryID, population) VALUES ('91346eb8-3dd2-11ed-b878-0242ac120002', 'Puebla', '83b3e642-3dd2-11ed-b878-0242ac120002', 3.2)
```

Entonces el programa podrá leer el archivo, correr cada comando secuencialmente y si hay una excepción como que por ejemplo los datos no estén con el formato correcto, el comando esté mal escrito o que se referencia a un país que no exista en la base de datos.

Entregas

1. **Requerimientos (15%)**
   Especifique los requerimientos a partir de lo que aparece en el enunciado de la tarea integradora.
   Entrega: Final de la semana 10.

2. **Implementación (70%) y diseño UML (15%) completos**
   Esta entrega debe tener la implementación funcionando perfectamente acorde a los requerimientos.
   Anexe el diseño, esta vez en su versión final. Este diseño tendrá nota a diferencia del diseño preliminar
   Entrega: Final de semana 13