



Tarea 3

Sistemas Operativos

Jorge Angel Juárez Vázquez
2213026247

Profesor: Jose Netz Romero Duran
30 August 2024

Tarea 3 - Hilos



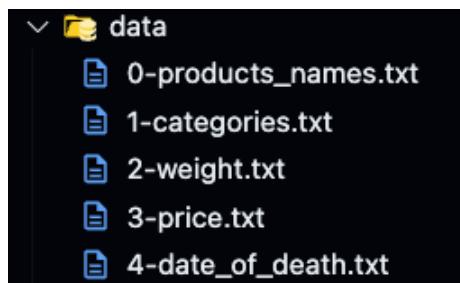
Negocio: WildFork

Lenguaje de programación: Python

Creación del archivo de negocio.

Para la creación del archivo que contiene los registros de los productos del negocio (WildFork) se siguió la siguiente secuencia:

1. Se crearon 5 archivos con información correspondiente a los atributos de cada registro, aproximadamente 50 valores por archivo.
 - a. 0-products_names.txt
 - b. 1-categories.txt
 - c. 2-weight.txt
 - d. 3-price.txt
 - e. 4-date_of_death.txt



2. Se creó un script para que el programa escogiera aleatoriamente un valor de cada uno de estos archivos y junto con un id secuencial (a partir de una cantidad especificada) creara los registros.

```

● ● ●

1 from random import choice
2 import sqlite3
3 from pathlib import Path
4 from typing import List, Generator
5
6
7 def get_id():
8     first_id = 2139
9     for _ in range(100000):
10         first_id += 1
11         yield first_id
12
13
14 def make_register(path: str, id_generator: Generator) -> List[str]:
15     register = []
16     data_files = Path(path).glob('*.txt')
17
18     register.append(next(id_generator))
19
20     for current_file in data_files:
21         lines = current_file.read_text(encoding='utf-8').splitlines()
22         register.append(choice(lines))
23
24     return register
25
26
27 if __name__ == '__main__':
28     registers_db = './data/WildFork.db'
29
30     id = get_id()
31
32     with sqlite3.connect(registers_db) as conn:
33         cursor = conn.cursor()
34         cursor.execute('''
35             CREATE TABLE IF NOT EXISTS registers (
36                 id INTEGER PRIMARY KEY,
37                 nombre_producto TEXT,
38                 categoria TEXT,
39                 peso_kg TEXT,
40                 precio TEXT,
41                 fecha_vencimiento TEXT
42             )
43         ''')
44
45     for _ in range(100000):
46         register = make_register('./data', id)
47         cursor.execute('''
48             INSERT INTO registers (id, nombre_producto, categoria, peso_kg, precio, fecha_vencimiento) VALUES (?, ?, ?, ?, ?, ?)'', tuple(register)
49     )

```

3. Finalmente con estos registros creados con valores aleatorios y un id secuencial, se incrustaron en una base de datos de SQLite.

	id	nombre_producto	categoria	peso_kg	precio	fecha_v...
1	2140	Brownies de Chocolate Fundente	Salsas y Aderezos	1.6	190	2024-04-10
2	2141	Pasta con Salsa	Brownies de Chocolate Fundente	1.5	356	2024-08-01
3	2142	Guiso de Verduras y Legumbres	Snacks y Aperitivos	0.3	149	2024-10-10
4	2143	Salmón Ahumado Deluxe	Mariscos y Pescados	0.5	190	2024-07-10
5	2144	Tacos de Carnitas Suaves	Sopas y Caldos	1	266	2024-05-15
6	2145	Galletas de Avena y Pasas	Aves y Pollos	1.4	113	2024-04-10
7	2146	Carne de Res Premium	Ensaladas y Guarniciones	2.8	33	2024-11-15
8	2147	Galletas de Avena y Pasas	Verduras Frescas	0.5	267	2024-11-10
9	2148	Ensalada de Quinoa y Verduras	Salsas y Aderezos	2.1	190	2024-04-01
10	2149	Pescado al Horno con Limón	Pasta y Granos	0.9	153	2024-05-01
11	2150	Tarta de Frutas de Temporada	Mariscos y Pescados	0.1	226	2024-05-01
12	2151	Tarta de Queso Clásica	Carnes Rojas	1.3	158	2024-02-30
13	2152	Pasta con Salsa Alfredo	Postres y Dulces	2.6	356	2024-08-20
14	2153	Guiso de Ternera Tradicional	Pasta y Granos	2.3	356	2024-03-10
15	2154	Brownies de Chocolate Fundente	Snacks y Aperitivos	2.5	341	2024-08-20
16	2155	Pechuga de Pavo a la Plancha	Verduras Frescas	0.2	145	2024-05-20
17	2156	Sushi de Salmón y Aguacate	Aves y Pollos	1.9	87	2024-11-20
18	2157	Tarta de Frutas de Temporada	Postres y Dulces	0.1	226	2024-12-20
19	2158	Sopa de Calabaza Especiada	Carnes Rojas	1.6	324	2024-10-10
20	2159	Arroz con Mariscos Exóticos	Mariscos y Pescados	2.5	12	2024-10-15
21	2160	Arroz con Mariscos Exóticos	Verduras Frescas	2.3	158	2024-02-14
22	2161	Tarta de Queso Especial	Mariscos y Pescados	2.4	397	2024-09-10
23	2162	Pescado Fresco del Día	Snacks y Aperitivos	1.6	208	2024-10-15
24	2163	Sopa de Mariscos Especial	Postres y Dulces	1.7	324	2024-09-20
25	2164	Tortillas de Maíz Caseras	Carnes Rojas	1.2	12	2024-12-01
26	2165	Pescado Empanizado Crujiente	Ensaladas y Guarniciones	2.1	158	2024-04-10

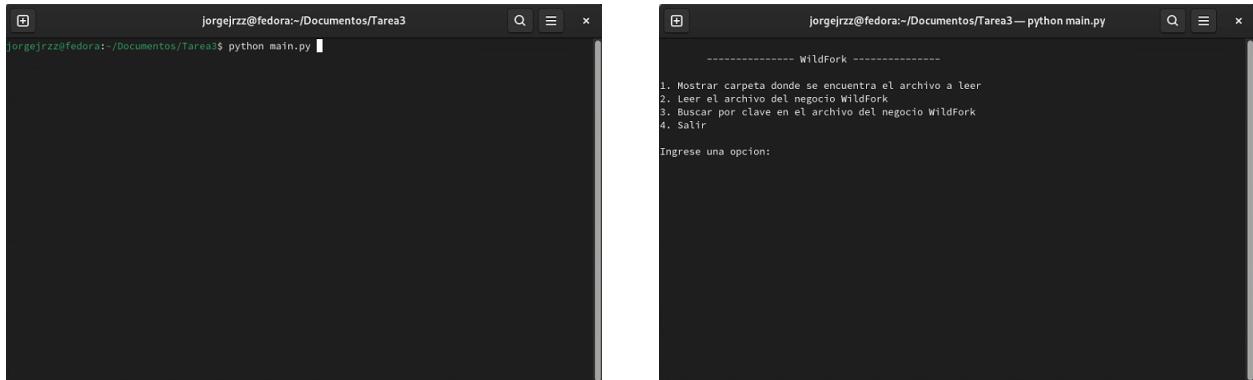
Opción 1 - Mostrar la carpeta en donde se encuentra el archivo a leer

Para la primera opción de menú, tenemos que hacer que nuestro programa ejecute por terminal el comando `ls -l`, esto se puede hacer con la diferentes instrucciones, pero para esta ocasión se optó por usar `subprocess.run()` la cual crea un proceso que puede ejecutar un programa o en este caso un comando por terminal, una característica importante es que no termina el proceso padre o principal.



```
1 # Ejecutar un proceso hijo con la ejecucion del comando `ls`  
2 result = subprocess.run(["ls", "-l"])  
3  
4 # Verificar si se ejecutó correctamente  
5 if result.returncode == 0:  
6     print("\n\n\tEl comando se ejecutó con éxito")  
7 else:  
8     print("\n\n\tHubo un error al ejecutar el comando")  
9  
10 input("\n\nPresione Enter para continuar.")
```

Ejecución del programa principal:



```
jorgejrzz@fedora:~/Documentos/Tarea3$ python main.py
```

```
jorgejrzz@fedora:~/Documentos/Tarea3— python main.py  
----- WildFork -----  
1. Mostrar carpeta donde se encuentra el archivo a leer  
2. Leer el archivo del negocio WildFork  
3. Buscar por clave en el archivo del negocio WildFork  
4. Salir  
Ingrese una opcion:
```

El código de la opción 1 genera la siguiente salida:

```
jorgejrzz@fedora:~/Documentos/Tarea3—python main.py

----- WildFork -----
1. Mostrar carpeta donde se encuentra el archivo a leer
2. Leer el archivo del negocio WildFork
3. Buscar por clave en el archivo del negocio WildFork
4. Salir

Ingrese una opcion: 1
total 4
-rw-r--r--. 1 jorgejrzz jorgejrzz    0 ago 31 00:09 codye.py
drwxrwxrwx. 1 jorgejrzz jorgejrzz  196 ago 30 21:29 data
-rw-rw-rw-. 1 jorgejrzz jorgejrzz 2602 ago 30 23:35 main.py
drwxrwxrwx. 1 jorgejrzz jorgejrzz    76 ago 30 22:53 src

El comando se ejecutó con éxito

Presione Enter para continuar.
```

Opción 2 - Leer el archivo del negocio WildFork

Para esta opción se creo otro directorio en donde van a estar los dos archivo que se vana a llamar desde el archivo principal, este directorio lleva por nombre `scr` y para esta opción del menú se creo un script el cual crea páginas de la lectura de la base de datos de 50 en 50 registros, dándole la opcional usuario de retroceder pagina, avanzar pagina y salir de la visualización de los datos, el siguiente es el código del archivo que hace posible esto:

```

● ● ●

1 import threading
2 import sqlite3
3 from queue import Queue
4 from typing import List
5 import os
6
7 def get_50_registers(db_path: str, page: int, queue: Queue):
8     with sqlite3.connect(db_path) as conn:
9         cursor = conn.cursor()
10        cursor.execute(f"SELECT * FROM registers LIMIT 50 OFFSET {((page - 1) * 50)}")
11        res = cursor.fetchall()
12
13    queue.put(res)
14
15    return res
16
17 def print_tuple_table(tuples: List):
18     # Definir los encabezados
19     headers = ["ID", "Nombre", "Categoria", "Peso (kg)", "Cantidad", "Fecha de Caducidad"]
20     # Convertir tuples a una lista de listas para facilitar el manejo
21     data = [list(t) for t in tuples]
22     # Encontrar el ancho máximo para cada columna
23     col_widths = [max(len(str(item)) for item in column) for column in zip(headers, *data)]
24     # Crear el formato de la línea
25     line_format = " | ".join("{:<" + str(width) + "}" for width in col_widths)
26     # Imprimir los encabezados
27     print(line_format.format(*headers))
28     # Imprimir una linea separadora
29     print("-" * (sum(col_widths) + 3 * (len(headers) - 1)))
30     # Imprimir cada fila de datos
31     for row in data:
32         print(line_format.format(*(str(item) for item in row)))
33
34
35 def show_db(db_path: str, page: int):
36     result_queue = Queue()
37     thread = threading.Thread(target=get_50_registers, args=(db_path, page, result_queue))
38     thread.start()
39     print("\n\t\t\t----- WildFork -----\\n\\n")
40     thread.join()
41     result = result_queue.get()
42     print_tuple_table(result)
43
44
45 def view_all(db_path: str):
46     page = 1
47     option = ''
48     while option != 'q':
49         thread = threading.Thread(target=show_db, args=(db_path, page))
50         os.system('clear')
51         thread.start()
52         thread.join()
53         print("\n\n-----")
54         print("Anteriores 50 registros: [b]\\tSiguientes 50 registros: [n]\\tSalir: [q]\\n")
55         option = input("Opcion: ").strip()
56         if option == 'n':
57             page += 1
58         elif option == 'b':
59             if page > 1:
60                 page -= 1
61
62
63 if __name__ == '__main__':
64     view_all('./data/WildFork.db')

```

Este archivo es llamado desde el archivo principal de la siguiente forma:

```
● ● ●

1 script = Path("./src/read.py")
2 # Ejecutar un proceso hijo con la ejecución del programa de mostrar los registros
3 result = subprocess.run(["python", script.absolute()])
4
5 # Verificar si se ejecutó correctamente
6 if result.returncode != 0:
7     print("\n\n\tHubo un error al ejecutar el comando")
8     input("\n\nPresione Enter para continuar.")
```

Estos códigos generan la siguiente salida por la línea de comandos:

ID	Nombre	Categoría	Peso (kg)	Cantidad	Fecha de Caducidad
2140	Brownies de Chocolate Fundente	Snacks y Aperitivos	1.6	198	2024-04-10
2141	Pasta con Salsa Pesto	Snacks y Aperitivos	1.5	356	2024-08-01
2142	Guiso de Verduras y Legumbres	Mariscos y Pescados	0.3	149	2024-10-10
2143	Salmon Alumado Deluxe	Sopas y Caldos	0.5	198	2024-07-10
2144	Tortilla de Pollo	Sopas y Caldos	1.0	246	2024-05-15
2145	Galletas de Avena y Pasas	Ensaladas y Guarniciones	1.4	113	2024-04-10
2146	Carne de Res Premium	Verduras Frescas	2.8	33	2024-11-15
2147	Galletas de Avena y Pasas	Snacks y Aperitivos	0.5	267	2024-04-10
2148	Ensalada de Queso y Verduras	Snacks y Aperitivos	2.0	199	2024-04-10
2149	Arroz con Pollo Maridado	Sopas y Caldos	1.9	153	2024-05-01
2150	Tarta de Frutas de Temporada	Carnes Rojas	0.1	226	2024-05-01
2151	Tarta de Queso Clásica	Carnes Rojas	1.3	158	2024-02-30
2152	Pasta con Salsa Alfredo	Postres y Dulces	2.6	356	2024-08-20
2153	Tortilla de Pollo con Queso Original	Aves y Pollos	2.0	356	2024-09-10
2154	Brownies de Chocolate Fundente	Verduras Frescas	1.5	141	2024-02-20
2155	Pechuga de Pavo a la Plancha	Aves y Pollos	1.9	208	2024-05-20
2156	Sushi de Salmon y Aguacate	Sopas y Caldos	0.2	33	2024-11-20
2157	Tarta de Queso de Temporada	Snacks y Aperitivos	1.0	145	2024-09-15
2158	Guiso de Calabaza Especiada	Mariscos y Pescados	1.4	87	2024-07-20
2159	Arroz con Mariscos Exóticos	Mariscos y Pescados	2.5	24	2024-10-10
2160	Arroz con Mariscos Exóticos	Mariscos y Pescados	2.3	324	2024-10-15
2161	Sopa de Mariscos Especial	Pasta y Granos	1.6	397	2024-09-10
2162	Pescado Fresco Especial	Aves y Pollos	1.5	32	2024-10-15
2163	Sopa de Mariscos Especial	Postres y Dulces	1.7	158	2024-09-01
2164	Tortillas de Maíz Caseras	Ensaladas y Guarniciones	1.2	226	2024-02-14
2165	Pescado Empalmado Crujiente	Postres y Dulces	2.1	324	2024-09-20
2166	Tarta de Limón Casera	Aves y Pollos	0.4	55	2024-04-10
2167	Ensalada de Queso Frescas	Carnes Rojas	1.5	356	2024-12-20
2168	Verduras Asadas al Horno	Postres y Dulces	1.1	113	2024-07-28
2169	Pechuga de Pollo Rellena	Snacks y Aperitivos	3	158	2024-03-10
2170	Pollo Orgánico Marinado	Snacks y Aperitivos	1.6	343	2024-04-20
2171	Brownies de Chocolate Fundente	Snacks y Aperitivos	2	358	2024-05-01
2172	Guisado Especial de Verduras	Ensaladas y Guarniciones	1.3	172	2024-08-15
2173	Hamburguesa de Res Artesanal	Pasta y Granos	1	231	2024-04-15
2174	Guiso de Verduras y Legumbres	Snacks y Aperitivos	1.8	231	2024-04-20
2175	Tacos de Pollo al Pastor	Pasta y Granos	2.2	368	2024-01-01
2176	Sopa de Calabaza Especiada	Aves y Pollos	0.7	158	2024-01-01
2177	Tortilla de Pollo con Pescado	Ensaladas y Guarniciones	0.7	207	2024-08-15
2178	Hamburguesa de Res Artesanal	Aves y Pollos	1.7	366	2024-02-20
2179	Tacos de Carnitas Suaves	Aves y Pollos	1.7	208	2024-07-10
2180	Tacos de Carnitas Suaves	Ensaladas y Guarniciones	2	230	2024-06-15
2181	Tacos de Pollo al Pastor	Carnes Rojas	1.5	369	2024-01-20
2182	Batido de Frutas Tropicales	Verduras Frescas	1.1	37	2024-01-01
2183	Pollo al Curry Suave	Verduras Frescas	0.6	87	2024-11-20
2184	Ensalada de Qutina y Verduras	Snacks y Aperitivos	2.7	359	2024-01-10
2185	Hamburguesa de Res Artesanal	Ensaladas y Guarniciones	0.7	70	2024-02-14
2186	Galletas de Avena y Pasas	Postres y Dulces	0.2	266	2024-01-15
2187	Ensalada de Queso Frescas	Ensaladas y Guarniciones	0.7	341	2024-02-28
2188	Pescado Fresco del Día	Postres y Dulces	0.8	267	2024-01-30
2189	Tarta de Frutas de Temporada	Aves y Pollos	1	55	2024-03-01

Anteriores 50 registros: [b] Siguientes 50 registros: [n] Salir: [q]
Opción: []

ID	Nombre	Categoría	Peso (kg)	Cantidad	Fecha de Caducidad
2190	Tacos de Carnitas Suaves	Pasta y Granos	1.1	324	2024-01-01
2191	Ensalada de Frutas Frescas	Carnes Rojas	3	267	2024-08-15
2192	Salmón a la Parrilla	Mariscos y Pescados	1.5	128	2024-01-20
2193	Sushi de Salmon y Aguacate	Mariscos y Pescados	1.6	324	2024-02-20
2194	Tortilla de Pollo con Queso	Postres y Dulces	1.6	370	2024-01-15
2195	Batido de Frutas Tropicales	Snacks y Aperitivos	0.2	358	2024-10-15
2196	Carne de Res Premium	Ensaladas y Guarniciones	2.1	30	2024-01-15
2197	Ensalada de Qutina y Verduras	Pasta y Granos	2.6	356	2024-12-10
2198	Cazuela de Pollo al Horno	Ensaladas y Guarniciones	2.5	266	2024-01-01
2199	Ensalada de Camarón Fresco	Ensaladas y Guarniciones	1.0	344	2024-10-10
2200	Pollo Orgánico Fresco	Verduras Frescas	0.2	86	2024-02-20
2201	Tortillas de Maíz Caseras	Postres y Dulces	0.9	277	2024-04-10
2202	Tortillas de Maíz Caseras	Snacks y Aperitivos	1.2	221	2024-06-20
2203	Pescado al Horno con Limón	Pasta y Granos	0.4	354	2024-08-20
2204	Tortilla de Lechuga Casera	Verduras Frescas	1.2	333	2024-03-03
2205	Hamburguesa de Res Artesanal	Snacks y Aperitivos	1.5	123	2024-05-20
2206	Pescado Fresco del Día	Verduras Frescas	2.6	386	2024-05-01
2207	Guiso de Pollo con Legumbres	Verduras Frescas	2.1	358	2024-09-20
2208	Tacos de Pollo al Pastor	Postres y Dulces	1.7	387	2024-01-15
2209	Tortilla de Pollo al Horno	Snacks y Aperitivos	2.6	226	2024-11-01
2210	Tacos de Pescado Fresco	Snacks y Aperitivos	2.2	86	2024-01-01
2211	Pescado al Horno con Limón	Snacks y Aperitivos	1.7	352	2024-11-15
2212	Pasta con Salsa Alfredo	Postres y Dulces	2.7	209	2024-06-20
2213	Tortilla de Pollo con Aperitivos	Postres y Dulces	3.0	268	2024-10-10
2214	Bravas de Chocolate Fundente	Snacks y Aperitivos	2.5	182	2024-11-15
2215	Ensalada César Clásica	Verduras Frescas	2.4	199	2024-07-01
2216	Pechuga de Pollo Rellena	Saladas y Aderezos	2.3	14	2024-08-01
2217	Tortilla de Espinacas Frescas	Saladas y Aderezos	1.7	341	2024-11-01
2218	Tortilla de Queso Frescas	Postres y Dulces	1.7	327	2024-02-15
2219	Pasta Integral Gourmet	Ensaladas y Guarniciones	0.4	190	2024-06-20
2220	Pasta con Salsa Pesto	Aves y Pollos	1.9	202	2024-02-10
2221	Galletas de Chocolate Chips	Postres y Dulces	0.2	236	2024-02-20
2222	Guisado Especial de Temporada	Postres y Dulces	1.4	368	2024-01-15
2223	Tortilla a la Barbacoa	Aves y Pollos	1	339	2024-11-01
2224	Pollo al Curry Suave	Aves y Pollos	3	359	2024-04-10
2225	Sopa de Calabaza Especiada	Mariscos y Pescados	0.2	37	2024-08-01
2226	Carne de Res Premium	Ensaladas y Guarniciones	2.3	221	2024-04-15
2227	Tarta de Queso Frescas	Snacks y Aperitivos	0.6	33	2024-07-20
2228	Arroz con Mariscos Exóticos	Postres y Dulces	1.5	328	2024-04-20
2229	Tarta de Limón Casera	Snacks y Aperitivos	3	266	2024-07-15
2230	Arroz con Mariscos Exóticos	Mariscos y Pescados	1.3	226	2024-11-01
2231	Pasta con Salsa de Tomate	Verduras Frescas	0.8	149	2024-04-15
2232	Tortilla a la Barbacoa	Ensaladas y Guarniciones	2.2	320	2024-06-15
2233	Pollo Orgánico Marinado	Ensaladas y Guarniciones	0.7	266	2024-04-18
2234	Pollo Orgánico Marinado	Mariscos y Pescados	0.6	123	2024-12-15
2235	Galletas de Avena y Pasas	Aves y Pollos	2.7	24	2024-04-20
2236	Hamburguesa de Res Artesanal	Saladas y Aderezos	1.4	37	2024-02-20
2237	Tortilla a la Barbacoa	Mariscos y Pescados	1.9	325	2024-03-15
2238	Ensalada de Frutas Frescas	Pasta y Granos	2.3	117	2024-01-20
2239	Arroz Frito con Verduras	Sopas y Caldos	2.5	33	2024-09-15

Anteriores 50 registros: [b] Siguientes 50 registros: [n] Salir: [q]
Opción: []

Opción 3 - Buscar por clave en el archivo del negocio WildFork

Para esta opción, se aprovechó la natural funcionalidad de una base de datos, por lo que no hubo dificultades en cuanto a la búsqueda de algún registro, simplemente se hizo la consulta, sin embargo igualmente que en la opción 3, se creó otro script dentro del directorio `scr`, el cual va a ser llamado como subproceso en el archivo principal, el siguiente es el contenido del archivo `search.py`:

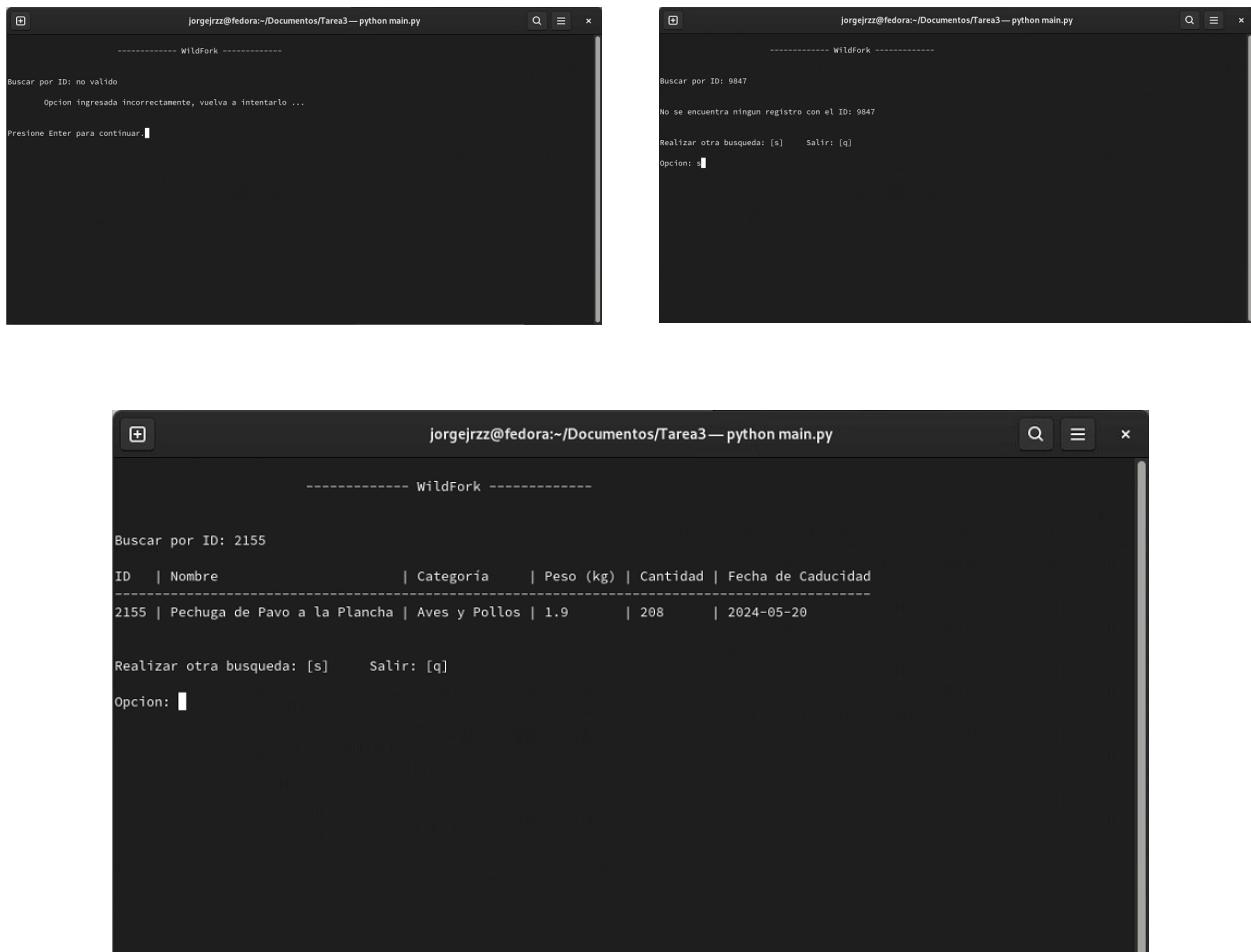


```
1 import os
2 import sqlite3
3 from queue import Queue
4 from threading import Thread
5
6
7 def serch_register(db_path: str, register_id: int, queue: Queue):
8     with sqlite3.connect(db_path) as conn:
9         cursor = conn.cursor()
10        cursor.execute(f"SELECT * FROM registers WHERE id = {register_id}")
11        res = cursor.fetchone()
12
13    queue.put(res)
14    return res
15
16 def print_tuple_table(tuple_data):
17     # Definir los encabezados
18     headers = ["ID", "Nombre", "Categoria", "Peso (kg)", "Cantidad", "Fecha de Caducidad"]
19     # Asegurarse de que tuple_data es una tupla
20     if not isinstance(tuple_data, tuple):
21         raise TypeError("El argumento debe ser una tupla")
22     # Verificar que la tupla tiene la longitud correcta
23     if len(tuple_data) != len(headers):
24         raise ValueError(f"La tupla debe tener {len(headers)} elementos")
25     # Convertir todos los elementos de la tupla a strings
26     data = [str(item) for item in tuple_data]
27     # Encontrar el ancho máximo para cada columna
28     col_widths = [max(len(header), len(item)) for header, item in zip(headers, data)]
29     # Crear el formato de la línea
30     line_format = " | ".join("{:<}" + str(width) + ")" for width in col_widths)
31     # Imprimir los encabezados
32     print(line_format.format(*headers))
33     # Imprimir una línea separadora
34     print("-" * (sum(col_widths) + 3 * (len(headers) - 1)))
35     # Imprimir la fila de datos
36     print(line_format.format(*data))
37
38 def user_serach(db_path: str):
39     result_queue = Queue()
40     option = ''
41     while option != 'q':
42         os.system('clear')
43         while True:
44             try:
45                 print("\n\t\t----- WildFork -----\\n\\n")
46                 id = int(input("Buscar por ID: ").strip())
47                 print()
48                 break
49             except ValueError:
50                 print("\nOpcion ingresada incorrectamente, vuelva a intentarlo ...")
51                 input("\nPresione Enter para continuar.")
52                 os.system('clear')
53
54             thread = Thread(target=serch_register, args=(db_path, id, result_queue))
55             thread.start()
56             thread.join()
57             result = result_queue.get()
58             if result is not None:
59                 thread = Thread(target=print_tuple_table, kwargs={'tuple_data': result})
60                 thread.start()
61                 thread.join()
62             else:
63                 print(f"\nNo se encuentra ningun registro con el ID: {id}")
64
65             print("\nRealizar otra busqueda: [s]\\tSalir: [q]\\n")
66             option = input("Opcion: ").strip()
67             if option == 'q':
68                 break
69
70
71 if __name__ == '__main__':
72     user_serach('./data/WildFork.db')
73
```

Este archivo es llamado desde el archivo principal de la siguiente forma:

```
● ● ●
1 script = Path("./src/search.py")
2 # Ejecutar un proceso hijo con la ejecución del programa de mostrar los registros
3 result = subprocess.run(["python", script.absolute()])
4
5 # Verificar si se ejecutó correctamente
6 if result.returncode != 0:
7     print("\n\n\tHubo un error al ejecutar el comando")
8     input("\n\nPresione Enter para continuar.")
```

Estos códigos generan la siguiente salida por la terminal de comandos:



The figure consists of three vertically stacked screenshots of a terminal window. The window has a dark background and a light-colored title bar. The title bar reads "jorgejrzz@fedora:~/Documentos/Tarea3 — python main.py". The window title is "WildFork".

- Screenshot 1:** Shows an invalid ID search. The output is:

```
----- WildFork -----
Buscar por ID: no valido
Opcion ingresada incorrectamente, vuela a intentarlo ...
Presione Enter para continuar: [ ]
```
- Screenshot 2:** Shows a non-existent ID search. The output is:

```
----- WildFork -----
Buscar por ID: 9847
No se encuentra ningun registro con el ID: 9847
Realizar otra búsqueda: [s] Salir: [q]
Opcion: [ ]
```
- Screenshot 3:** Shows a successful search for ID 2155. The output is:

```
----- WildFork -----
Buscar por ID: 2155
ID | Nombre | Categoría | Peso (kg) | Cantidad | Fecha de Caducidad
---+-----+-----+-----+-----+-----+
2155 | Pechuga de Pavo a la Plancha | Aves y Pollos | 1.9 | 208 | 2024-05-20
Realizar otra búsqueda: [s] Salir: [q]
Opcion: [ ]
```

Opción 4 - Salir

Finalmente, la opción 4 que corresponde a terminar con la ejecución del programa, es creada con el siguiente bloque de código y genera lo siguiente por la terminal:



```
1 os.system("clear")
2 print("\n\tVuleva pronto <3\n")
3 break
```

The screenshot shows a terminal window titled "jorgejrzz@fedora:~/Documentos/Tarea3". The window contains the following text:
Vuleva pronto <3
jorgejrzz@fedora:~/Documentos/Tarea3\$

Archivo [main.py](#):



```
1 import os
2 from pathlib import Path
3 import subprocess
4 from threading import Thread
5
6 def menu() -> int:
7     os.system('clear')
8     print("\n\t----- WildFork -----")
9     print("1. Mostrar carpeta donde se encuentra el archivo a leer")
10    print("2. Leer el archivo del negocio WildFork")
11    print("3. Buscar por clave en el archivo del negocio WildFork")
12    print("4. Salir")
13    try:
14        option = int(input("\nIngrese una opcion: "))
15        return option
16    except ValueError:
17        print("\n\tOpcion ingresada incorrectamente, vuelva a intentarlo ... ")
18        input("\n\nPresione Enter para continuar.")
19        os.system('clear')
20        menu()
21
22
23 def main():
24    while True:
25        option = menu()
26        match option:
27            case 1:
28                # Ejecutar un proceso hijo con la ejecucion del comando `ls`
29                result = subprocess.run(["ls", "-l"])
30
31                # Verificar si se ejecutó correctamente
32                if result.returncode == 0:
33                    print("\n\tEl comando se ejecutó con éxito")
34                else:
35                    print("\n\tHubo un error al ejecutar el comando")
36
37                input("\n\nPresione Enter para continuar.")
38
39            case 2:
40                script = Path("./src/read.py")
41                # Ejecutar un proceso hijo con la ejecucion del programa de mostrar los registros
42                result = subprocess.run(["python", script.absolute()])
43
44                # Verificar si se ejecutó correctamente
45                if result.returncode != 0:
46                    print("\n\tHubo un error al ejecutar el comando")
47                    input("\n\nPresione Enter para continuar.")
48
49            case 3:
50                script = Path("./src/search.py")
51                # Ejecutar un proceso hijo con la ejecucion del programa de mostrar los registros
52                result = subprocess.run(["python", script.absolute()])
53
54                # Verificar si se ejecutó correctamente
55                if result.returncode != 0:
56                    print("\n\tHubo un error al ejecutar el comando")
57                    input("\n\nPresione Enter para continuar.")
58
59            case 4:
60                os.system("clear")
61                print("\n\tVuela pronto <3\n")
62                break
63            case _:
64                print("Esa opcion no se encuentra en el menu :(")
65                input("\n\nPresione Enter para continuar.")
66
67
68 if __name__ == '__main__':
69     main()
```

Como se puede observar, se empleo en diferentes tareas y para diferentes propósitos el uso de Hilos y nuevamente de procesos.

Conclusiones

La realización de esta práctica ha proporcionado una valiosa experiencia en el manejo de procesos y threads en sistemas operativos, así como en la manipulación de archivos y la implementación de estructuras de datos en un contexto práctico. Los principales puntos a destacar son:

1. Manejo de procesos: La implementación del programa principal y la invocación de subprogramas mediante subprocess ha reforzado la comprensión de cómo los sistemas operativos gestionan y ejecutan múltiples procesos.
2. Manipulación de archivos: La creación y lectura de un archivo con 100,000 registros aleatorios ha permitido practicar operaciones de entrada/salida a gran escala, esenciales en muchas aplicaciones del mundo real.
3. Programación modular: La división del programa en diferentes componentes (principal, leeArchivo, buscaRegistro) ha reforzado la importancia de la modularidad en el desarrollo de software.
4. Aplicación práctica: El proyecto ha simulado un escenario de negocio real, permitiendo ver cómo los conceptos teóricos de sistemas operativos se aplican en situaciones prácticas.

Finalmente el uso de Python para esta practica creo que fue una gran elección ya que cuenta con diversos módulos nativos que facilitaron el desarrollo de esta practica.