Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Laboratorio de Lenguajes Formales y de Programación

Ing. Otto Amílcar Rodríguez Acosta

Aux. Javier Alberto Cabrera Puente

SimpleQL

Manual Técnico

Romeo Ernesto Marroquín Sánchez

Carné: 201902157

Fecha: 30/08/2020

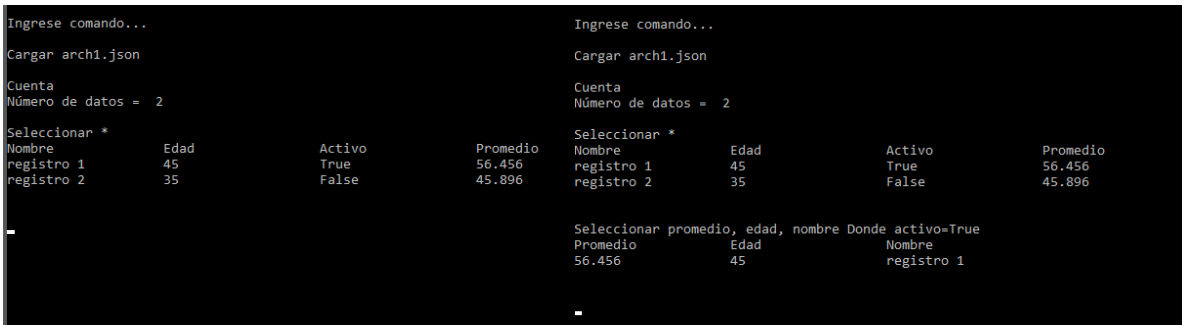
Flujo del código de la Aplicación

La aplicación lee los comandos ingresados por el usuario a través de un ciclo WHILE, al momento de iniciar el programa se da la instrucción de ingresar un comando, y el programa queda a la escucha para ingresar cualquier comando, para salir de este bucle WHILE el comando a utilizar es CERRAR, sin este comando la consola seguirá leyendo el teclado del usuario.

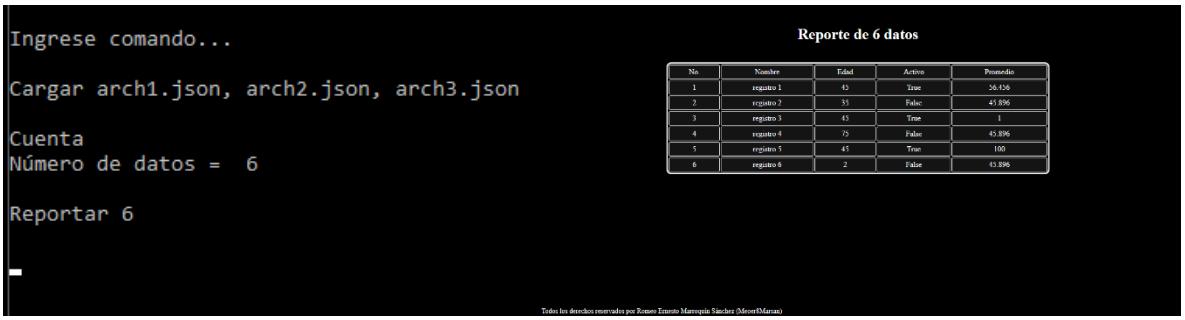
Al momento de ingresar cualquier comando, el programa entra al ciclo WHILE, y empieza a comparar la primera palabra del comando ingresado, esto convirtiendo primero toda la cadena a minúsculas para que no importe como el usuario haya ingresado el comando (case insensitive), las comparaciones son para ver exactamente qué comando (de los 7 disponibles) está utilizando.

Luego de interpretar qué comando ingresó el usuario, el programa procede a “splittear” todos los componentes de ese comando, convirtiéndolo en una lista de cadenas String para su posterior análisis, esto aplica para todos los comandos que necesitan parámetros de varios campos (CARGAR y SELECCIONAR), ya que los demás comandos tienen parámetros ya definidos, o sólo pueden tomar ciertos valores, por lo que no puede haber problemas con su interpretación.

El análisis que se realiza para el comando CARGAR no es muy complejo, debido a que solo se debe “splittear” todos los nombres de los archivos .json ingresados (y separados por coma) y cargarlos a memoria utilizando la librería json, por lo que no tiene mucha complicación. Ahora bien, el comando seleccionar fue dividido en dos secciones, cuando se selecciona todo (*) y cuando se selecciona por campos (nombre, edad, etc…), y cada uno de estos a su vez, se subdivide si utiliza la posibilidad de restricción (DONDE) o no, por lo que en total se tienen 4 tipos diferentes del comando SELECCIONAR, y se interpretan de diferente manera, siendo el más simple de ellos el seleccionar todo sin restricción (SELECCIONAR *) y siendo el más complejo de analizar el que utiliza el comando por campos en diferentes órdenes y utiliza la restricción (SELECCIONAR promedio, edad, nombre DONDE activo=True)



Para los comandos sencillos como SUMA, CUENTA, MAXIMO y MINIMO lo único que se hace es recorrer la lista con todos los registros, haciendo la operación correspondiente para cada comando, ya sea comparaciones, sumas o simplemente autoincrementando un valor inicial 0, por lo que no poseen gran complejidad de su estructura en código. Esto a excepción del comando REPORTAR n, debido a que éste no sólo recorre la lista a través de todo sus registros y campos, sino que además realiza la codificación en HTML de estos campos y agrega un estilo en CSS a la misma página. Para explicar esto, primero debemos irnos al recorrido, el cual es un FOR el cual recorre toda la lista actual de datos, y en cada uno de los parámetros se concatenan comandos HTML para la generación de una tabla (variable 1), luego se crea una variable para almacenar todo el código HTML básico (variable 2), por último se concatenan cada una de las variables de forma lógica para formar el código HTML de la página, y se guarda en el dispositivo utilizando la extensión .html, una vez almacenado el archivo se utiliza una librería importada llamada webbrowser el cual posee el comando “open_new_tab” el cual abre una página web determinada, en este caso la tabla generada anteriormente, por lo que la página se abre automáticamente (sin cerrar la consola) generando un reporte HTML con el siguiente formato:



Para finalizar con el ciclo WHILE basta con escribir el comando CERRAR desde la consola, y ésta se cerrará automáticamente, y TODOS LOS DATOS almacenados en la tabla se perderán permanentemente, por lo que para volver a utilizarlos se debe volver a ejecutar el programa y volver a cargar todos los archivos .json, por eso es recomendable verificar dos veces antes de utilizar el comando CERRAR.

Nota Especial: El programa no tiene reconocimiento de errores FUNCIONALES, por lo que si se comete algún error al momento de escribir algún campo, se escribe mal algún nombre de un archivo .json, o con la gestión de algún registro en particular el programa se cerrará automáticamente, por lo que se debe verificar que cada campo sea ingresado correctamente (considerando que el comando está bien escrito). Ahora bien si se ejecuta un comando inexistente, el programa simplemente no hará nada, y volverá a leer el teclado para la ejecución de otro comando.

Especificaciones Mínimas y Recomendaciones

Para correr el programa de manera óptima se deben contar con 17 KB de memoria LIBRE en su dispositivo, esto mas el espacio de almacenamiento que ocupen sus archivos .json y los datos que se requieran procesar, (esto también varía el tamaño del archivo .html del reporte) por lo que una cantidad de memoria óptima sería de 50 KB para trabajar de manera normal con tablas de datos normales (no bases de datos empresariales).

El dispositivo que ejecute el programa debe tener un hilo de ejecución disponible debido a que la aplicación es secuencial por bloques, por lo que sólo necesita un hilo del procesador para poder funcionar, (esto no afecta la labor de otras tareas a menos que el procesador esté generando un efecto de cuello de botella por otras aplicaciones).