**DOCUMENTACIÓN DE LAS FUNCIONALIDADES**

**PROYECTO: ISABELLA HERNÁNDEZ OCHOA**

Modificaciones al struct Local:

* Se modifica el campo estado el cual permite identificar si el local está ocupado o disponible, este en la versión pasada era numérico ahora se maneja con enum.
* Se añade el campo numTrabajadores, valorAlquiler, ventas y valorConsumo.

Procedimiento *modificarLocal*: Permite modificar varios campos de la información del local como nombreLocal, promedio de consumo de servicios y ventas del local, en la versión anterior solo dejaba modificar el campo nombreLocal.

* Entradas: Matriz y las dimensiones de esta, filas y columnas.
* Salida: Ninguna.
* Conceptos: Manejo de memoria dinámica, matriz con structs, uso de condicionales, uso de structs y enums.

Procedimiento *buscarLocalNombre*: Nueva funcionalidad que quería realizar en la versión anterior y no la realice, este procedimiento le permite al usuario Visitante a partir del nombre del local buscar información de la ubicación del local.

* Entradas: Matriz y las dimensiones de esta, filas y columnas.
* Salida: Ninguna.
* Conceptos: Manejo de memoria dinámica, matriz con structs y bucle for.

Función *calcularOcupados*: Retorna la cantidad de locales ocupados en el piso ingresado por el usuario, este valor será el tamaño de la lista que realizará el proceso de ordenamiento por medio de algoritmos como QuickSort y MergeSort.

* Entradas: Matriz del centro comercial, el número de locales por piso del centro comercial y el número del piso que ingreso el usuario.
* Salida: Cantidad de locales ocupados.
* Conceptos: Manejo de memoria dinámica, matriz con structs, acumuladores y bucle for.

Función *calcularOcupadosCC*: Retorna la cantidad de locales ocupados en **toda** la matriz del centro comercial, este valor será el tamaño de la lista que realizará el proceso de ordenamiento por medio de algoritmos iterativos como el de Selección e Inserción.

* Entradas: Matriz del centro comercial, el número de locales por piso del centro comercial y el número del piso que ingreso el usuario.
* Salida: Cantidad de locales ocupados.
* Conceptos: Manejo de memoria dinámica, matriz con structs, acumuladores y bucle for.

Procedimiento *ordenarValorAlquiler*: Nueva funcionalidad que haciendo el llamado del algoritmo recursivo MergeSort permite ordenar de menor a mayor el valor de alquiler de los locales del piso ingresado por el usuario.

* Entradas: Matriz y las dimensiones de esta, filas y columnas.
* Salida: Ninguna.
* Conceptos: Manejo de memoria dinámica, matriz con structs, lista con struct, y manejo del algoritmo recursivo MergeSort.

Procedimiento *ordenarNumeroTrabajadores*: Nueva funcionalidad que haciendo el llamado del algoritmo recursivo QuickSort permite ordenar de menor a mayor el número de trabajadores de cada local teniendo en cuenta el piso del centro comercial ingresado por el usuario.

* Entradas: Matriz y las dimensiones de esta, filas y columnas.
* Salida: Ninguna.
* Conceptos: Manejo de memoria dinámica, matriz con structs, lista con struct y manejo del algoritmo QuickSort.

Procedimiento *imprimirOrdenados:* Procedimiento que se encarga de imprimir las listas ordenadas al hacer uso de los diferentes algoritmos de ordenamiento para organizar el piso o todo el centro comercial teniendo en cuenta el valor de ventas, cantidad de empleados, valor del local y el valor de consumo de servicios de los locales.

* Entradas: Lista de struct y el tamaño de dicha lista.
* Salida: Ninguna.
* Conceptos: Conceptos básicos y estructura for.