****



**“Muster Cloud”**

**YONATHAN ALEXANDER HERNÁNDEZ SATZ 201900619**

**GUATEMALA, 15 DICIEMBRE DEL 2022**

**Tabla de contenido**

**Introducción …………………………………………………………………………………………………………………… 3**

**Información destacada …………………………………………………………………………………………….…………. 3**

**Objetivos ………………………………………………………………………………………………………. 3**

**1.Requerimientos ………………………………………………………………………………………………………………. 4**

**2.Instalación de aplicaciones …………………………………………………………………………………..………. 5**

**2.1 Instalación de Visual Studio Code**

**3. Configuración de sistema ………………………………………………………………………………………………. 6**

**4. Estructura raíz ……………………………………………………………………………………………..…………..…… 7**

**4.1.1 App.js**

**4.1.2 Index.html**

**4.1.3 Style.css**

**4.1.3 Archivos de recursos .json**

**5. Estructura ……………………………………………………………………………………………………………………... 9**

**5.1.2 Lista Simple**

**5.1.3 Lista de listas**

**5.1.4 Lista circular doblemente enlazada**

**5.1.5 Pila**

**5.1.6 Cola**

**5.1.7 Árbol**

**5.1.8 Matriz dispersa**

**6. Funciones generales ………………………………………………………………………………..……………..…14**

**7. Consideraciones generales …………………………………………………………………………………….... 16**

**8. Flujo de análisis de un Json …………………………………………………………………………………..……… 16**

**Introducción**

El presente documento describe los aspectos técnicos informáticos del sistema de información. El documento introducirá al personal técnico especializado encargado de las actividades de mantenimiento, revisión, solución de problemas, instalación y configuración del sistema.

**Información destacada**

El manual técnico hace referencia a la información necesaria con el fin de orientar al personal en la concepción, planteamiento análisis programación e instalación del sistema. Es de notar que la redacción propia del manual técnico está orientada a personal con conocimientos en sistemas y tecnologías de información, conocimientos de programación media/avanzada en programación orientada a estructuras de datos y desarrollo en el entorno del lenguaje de programación **JAVASCRIPT.**

**Objetivos**

Instruir el uso adecuado de **Muster Cloud**, para el acceso oportuno y adecuado en la instalación de este, mostrando los pasos a seguir en el proceso de instalación, así como la descripción de los archivos relevantes del sistema los cuales nos orienten en la configuración y soporte de este.

1. **Requerimientos**

El sistema pude ser abierto en cualquier sistema operativo que cumpla con los siguientes requerimientos (se recomienda el uso de Windows) :

**Software**

* Microsoft Visual Studio (cualquiera de sus ediciones)

**Hardware**

* Equipo con al menos 2 GB RAM
* Equipo con al menos 4 GB disponible en el disco duro.

1. **Instalación de aplicaciones**

Para implementar de manera correcta **Muster Cloud** en un sistema es necesario instalar ciertos programas, los cuales no necesitan de una conexión a internet de forma obligatoria.

* 1. **Instalación Visual Code**

Para su respectiva instalación nos dirigimos en el navegador a :

<https://code.visualstudio.com/>

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

La página nos desplegara una vista intuitiva donde se procederá a presionar el botón **AZUL (**Download for Windows)

Icono

Descripción generada automáticamenteSe descargara un archivo con el nombre VSCodeUserSetup , el cual procederemos a abrir e instalar en nuestra disco local.

1. **Configuración del sistema**

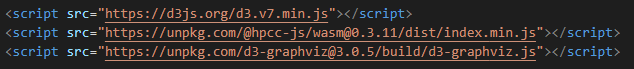
**Muster Cloud** está desarrollado bajo JavaScript, esto permite estructurar el sistema de manera rápida lo que se facilita el mantenimiento a dicha solución, a continuación, se describe la estructura básica del sistema y se enfatiza en los archivos, librerías y directorios relevantes para su configuración y adaptación.

1. **Bibliotecas / Recursos**
   1. **D3**

**¿Qué es D3?**

D3 son las siglas de Data-Driven Documents. Es una biblioteca JavaScript de código abierto desarrollada por Mike Bostock para crear visualizaciones de datos interactivas personalizadas en el navegador web utilizando SVG, HTML y CSS.

D3 será utilizado para renderizar el código DOT generado en cada estructura para su posterior visualización. Para la implementación de D3 basta con agregar las siguientes líneas de código en el body del index.



1. **Estructura Raíz**

El proyecto “**Muster Cloud**” tiene la siguiente estructura de directorios:

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

A continuación, se describirán los directorios y archivos más importantes:

* + 1. **App.js**

Incluye archivos que contienen declaraciones y definiciones de macro que se compartirán entre varios archivos de origen. Se llamarán a los archivos de origen haciendo referencia a su respectivo nombre.

* + 1. **Index.html**

Es la página por defecto dentro de los directorios del sitio web. En este caso el propio servidor web es el que se encarga de buscar el archivo index.html y mostrarlo al visitante (Contiene la interfaz de usuario)

* + 1. **Style.css**

CSS (hojas de estilo en cascada), incluirá archivos codificados que seleccionan elementos del index y controlan su presentación.

.

* + 1. **Archivos de recursos. json**

Son archivos de prueba, que pueden ser cargados en el sistema para revisar el correcto funcionamiento de cada apartado del usuario/administrador.

**5. Estructura**

* 1. **Archivo de origen**

**5.1.1 Funcionamiento de las estructuras de datos**

**5.1.2 Lista Simple**

|  |  |
| --- | --- |
| **Class Node** | Inicializa un nodo con un constructor, parámetros (dpi,name,username,password,phone,admin)  posteriormente |
| **Class ListaSimple** | Contiene un constructor con una cabeza y un tamaño para el recorrido de la lista. |
| **push** | Inserta un nodo al frente de la lista |
| **append** | Inserta un nodo al final de la lista |
| **registrar** | Obtiene los valores ingresados en el apartado de registrarse para crear un nuevo nodo con esa información. |
| **deleteNodo** | Elimina un nodo seleccionado. |
| **getCount** | Obtiene el tamaño de la lista simple. |
| **buscar** | Busca un nodo en especifico, pasando el valor como parámetro |
| **printList** | Imprime en consola el recorrido de la lista simple |
| **graficarLista** | Obtiene el recorrido de la lista y lo genera en un .dot |
| **mostrarUsuarios** | Obtiene los usuarios de la lista simple |
| **enviarUsuarios** | Envia la información hacia una lista para su posterior uso |

**5.1.3 Lista de listas**

|  |  |
| --- | --- |
| **Class Nodo** | Contiene un constructor con un dato, un siguiente , anterior y una lista simple |
| **Class Lista** | Contiene un constructor con un contador para saber el numero de elementos y un primero haciendo referencia a la cabeza de la segunda lista. |
| **add** | Crea una lista normal . |
| **add2** | Con parámetros nombre,lista : ingresa una nueva a una ya creada. |
| **repetido** | Verifica si en la lista normal existe un nodo repetido. |
| **mostrar** | Imprime en consola el recorrido de las listas |
| **buscar** | Dado un índice busca la información en la lista. |
| **graficarListaDeListas** | Obtiene el recorrido de la lista y lo genera en un .dot |
| **mostrarArtistas** | Obtiene los usuarios de la lista de listas |
| **enviarArtistas** | Envia la información hacia una lista de artistas para su posterior uso |
| **ordenar2** | Ordena la lista de manera ascendente usando el método Quicksort. |

**5.1.4 Lista circular doblemente enlazada**

|  |  |
| --- | --- |
| **NodeDoubleList** | Inicializa un nodo con un constructor, parámetro “valor”, un siguiente y un anterior. Esto para recorrer posteriormente la lista |
| **Class CircularDoublyLinkedList** | Contiene un constructor con parámetro “valor”, una cabeza, una cola y lenght (para la dimensión) |
| **initialize** | Inicializa un nodo con un valor. |
| **append** | Inserta un valor al inicio de la lista. |
| **prepend** | Inserta un valor al final de la lista. |
| **toArray** | Recorre la lista y lo imprime con consola. |
| **insert** | Inserta un valor en una posición especifica. |
| **deleteHead** | Borra el valor de la cabeza de la lista. |
| **deleteTail** | Borra el valor de la cola de la lista. |
| **delete** | Elimina un valor en especifico por medio de un index. |
| **buscarCircular** | Busca un nodo por medio de un index. |
| **graficarlistaCircular** | Obtiene el recorrido de la lista y lo genera en un .dot |

**5.1.5 Pila**

|  |  |
| --- | --- |
| **Class nodoPila** | Inicializa un nodo con un constructor, parámetros valor y siguientes. |
| **Class Stack** | Contiene un constructor con una cabeza. |
| **push** | Ingresa un valor a la cima de la pila. |
| **pop** | Elimina el primer valor de la pila (cima). |
| **display** | Recorre la pila y lo muestra en consola. |
| **mostrarAmigos** | Obtiene los amigos de pila (nodos). |
| **enviarAmigos** | Envia la información hacia una pila de amigos. |
| **graficarpila** | Obtiene el recorrido de la pila y lo genera en un .dot |

**5.1.6 Cola**

|  |  |
| --- | --- |
| **Class nodoCola** | Inicializa un nodo con un constructor, parámetros info y siguiente |
| **Class cola** | Contiene un constructor con un primero, un último y un tamaño. |
| **insertar** | Ingresa un valor al final de la cola. |
| **mostrar** | Recorre toda la cola y la imprime en consola. |
| **getTheIndex** | Obtiene la posición de un nodo en específico. |
| **eliminarPrimero** | Elimina el primer elemento de la cola. |
| **graficarCola** | Obtiene el recorrido de la cola y lo genera en un .dot |

**5.1.7 Árbol**

|  |  |
| --- | --- |
| **Class nodoArbol** | Inicializa un nodo con un constructor, parámetros datos,izquierda,derecha |
| **Class Arbol** | Contiene un constructor con una raiz |
| **agregar\_recursivo** | Añade un elemento recorriendo de manera recursiva el árbol. |
| **inorden\_recursivo** | Recorre el árbol para colocar los elementos de manera , inorder. |
| **buscar** | Busca un nodo en el árbol. |
| **agregar** | Agrega un elemento a el recorrido recursivo. |
| **buscar2** | Busca un nodo en especifico en el árbol. |
| **graficarArbol** | Obtiene el recorrido del árbol y lo genera en un .dot |

**5.1.7 Matriz Dispersa**

|  |  |
| --- | --- |
| **Class nodoMatrizDispersa** | Inicializa un nodo con parámetros, dato, posVertical,posHorizontal, derecha,izquierda,arriba,abajo |
| **Class MatrizDispersa** | Contiene un constructor con tres raíces, haciendo referencia a la posición vertical, horizontal y un NodoDispersa |
| **crearIndiceVertical** | Crea un índice vertical. |
| **crearIndiceHorizontal** | Crea un índice horizontal. |
| **insertarVertical** | Inserta a la matriz un índice vertical. |
| **insertarHorizontal** | Inserta a la matriz un índice horizontal. |
| **insertarDato** | Inserta un dato a la matriz dispersa. |
| **insertarDatoDispersa** | Inserta un dato en las posiciones especificadas (vertical , horizontal) |
| **recorrerMatriz** | Obtiene el recorrido de la matriz y lo genera en un .dot |

**6. Funciones generales**

|  |  |
| --- | --- |
| **login** | Valida las credenciales de un usuario. |
| **logout** | Cierra la sesión devolviendo el estado de la web a div iniciales. |
| **CargaMasivaUsuarios** | Carga al sistema usuarios por medio de un archivo json. |
| **ordenamiento** | Ordena los artistas de manera ascendente con el método burbuja. |
| **CargaMasivaArtistas** | Carga al sistema artistas por medio de un archivo json. |
| **CargaMasivaCanciones** | Carga al sistema canciones por medio de un archivo json. |
| **CargaMasivaPodcast** | Carga al sistema podcasts por medio de un archivo json. |
| **CargaMasivaMusicaProgramadas** | Carga al sistema música programada por medio de un archivo json. |
| **agregarOptionPlaylist** | Agrega a la playlist la opción seleccionada en el select. |
| **cancionActual** | Devuelve la canción inicial en la playlist. |
| **siguienteCancion** | Recorre la lista circular (playlist) hacia la siguiente posición. |
| **anteriorCancion** | Recorre la lista circular (playlist) hacia la anterior posición. |
| **agregarOptionPila** | Agrega la opción seleccionada en el select a la pila de amigos. |
| **eliminarAmigo** | Elimina la cima de la pila de amigos. |
| **agregarOptionColaBloqueo** | Agrega la opción seleccionada a la cola de bloqueos. |
| **desbloquearUsuarioCola** | Elimina el ultimo elemento de la cola de bloqueos. |
| **borrarOptionsListaUsuarios** | Limpia las opciones del select de usuarios. |
| **borrarOptionsPlaylist** | Limpia las opciones del select de música. |
| **publicarInmediatamente** | Publica una canción en la lista de listas. |
| **publicarProgramado** | Publica una canción en la matriz dispersa en las posiciones indicadas. |
| **limpiarFormularios** | Limpia formularios después de su uso. |
| **showDivIniciales** | Muestra los divs iniciales (ingreso a la web) |
| **showDivAdministrador** | Muestra los divs correspondientes a un usuario de tipo administrador. |
| **showDivUsuario** | Muestra los divs correspondientes a un usuario de tipo común. |

1. **Consideraciones de clases, directorios y archivos:**

Todos los archivos y directorios no mencionados son parte importante para el funcionamiento del sistema, no se hacen referencia en este documento debido a que solo se enfatizan los archivos que el usuario puede en un dado caso modificar, con conocimiento previo de lo que se hace.

* No se puede analizar un archivo sin antes cargarlo al sistema.
* Es importante fijarse en el formato de los archivos de prueba. json del sistema, ya que ese formato será el aceptado en el sistema.
* El sistema no cuenta con persistencia de datos. El recargar la página hará que pierda su información.
* Las estructuras de datos se comparten entre todos los usuarios.

1. **Flujo del análisis de Json**

|  |
| --- |
| **Nombre: Análisis de un archivo.** |
| **Actor(es):** Administrador |
| **Descripción:** Se analiza un archivo. json cargado al sistema en busca de diferente información encapsulada por partes (usuarios,tutorial,articulos) |
| **Flujo Normal:**   1. El actor ingresa a la web y elige la opción de carga. 2. Se carga la información y se añade a sus respectivas estructuras. 3. Se analiza la información para ser procesada. 4. Se genera en la opción de reportes su visualización. |
| **Flujo alternativo:**   1. Error en la lectura del archivo |
| **Post condición:**   1. Archivos gráficos/información solicitada. |