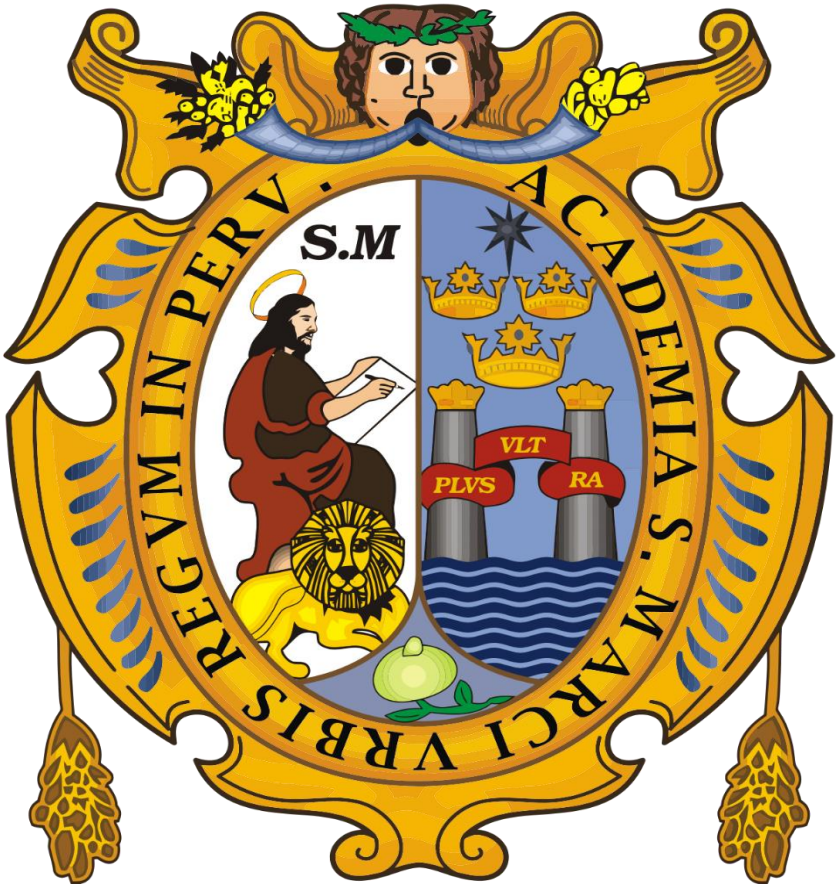
“Año de la universalización de la salud”

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS



FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMACIÓN

ANALSIS Y DISEÑO DE ALGORITMOS

TEMA: PROYECTO FINAL

INTEGRANTES:

- ✓ Sifuentes Marcelo Roberto
- ✓ Silva Gomez Giancarlo
- ✓ Vera Cueva Antonella
- ✓ Huallpa Chirinos Moisés
- ✓ Coronel Ramírez Patrick Miguel

LIMA – PERÚ

2020

INTRODUCCIÓN

A lo largo del curso de Análisis y Diseño de Algoritmos hemos visto distintos algoritmos que realizan funciones específicas como son de ordenamiento, búsqueda, selección, entre otros. Dentro de estas funciones que realizan los algoritmos; pudimos ver que se pueden diseñar 2 o más algoritmos distintos que realicen la misma función. En el caso de los algoritmos de ordenación existe el Bubble Sort, Merge Sort, Counting Sort, etc. Cada uno con una aproximación diferente y un modo de resolver distinto. Por supuesto al existir diversos algoritmos que permitan resolver 1 problema en particular, la elección de un algoritmo en particular dependerá de diversos factores como lo son el tiempo de ejecución del algoritmo, los recursos tecnológicos e incluso la naturaleza del problema a resolver. En clase se ha analizado con mayor profundidad cada algoritmo, así que en este informe más bien se buscará mostrar de una manera gráfica y práctica el desempeño de algunos algoritmos de ordenación como lo son: **Selection Sort, Quick Sort, Counting Sort** a través de una aplicación web realizada con JavaScript, HTML y CSS por los integrantes del grupo que aparecen en la caratula del presente informe.

MARCO TEÓRICO

Primero realizaremos un repaso rápido de los algoritmos que se nos dejó para realizar la aplicación web los cuales fueron mencionados en la introducción.

1. **Selection Sort**

El algoritmo de Selection Sort ordena un arreglo encontrando repetidamente el menor elemento de la parte aún no sorteada y poniéndolo al inicio. Este algoritmo mantiene 2 sub arreglos en un arreglo dado:

- El sub arreglo que está ordenado.
- El sub arreglo restante que todavía no está ordenado.

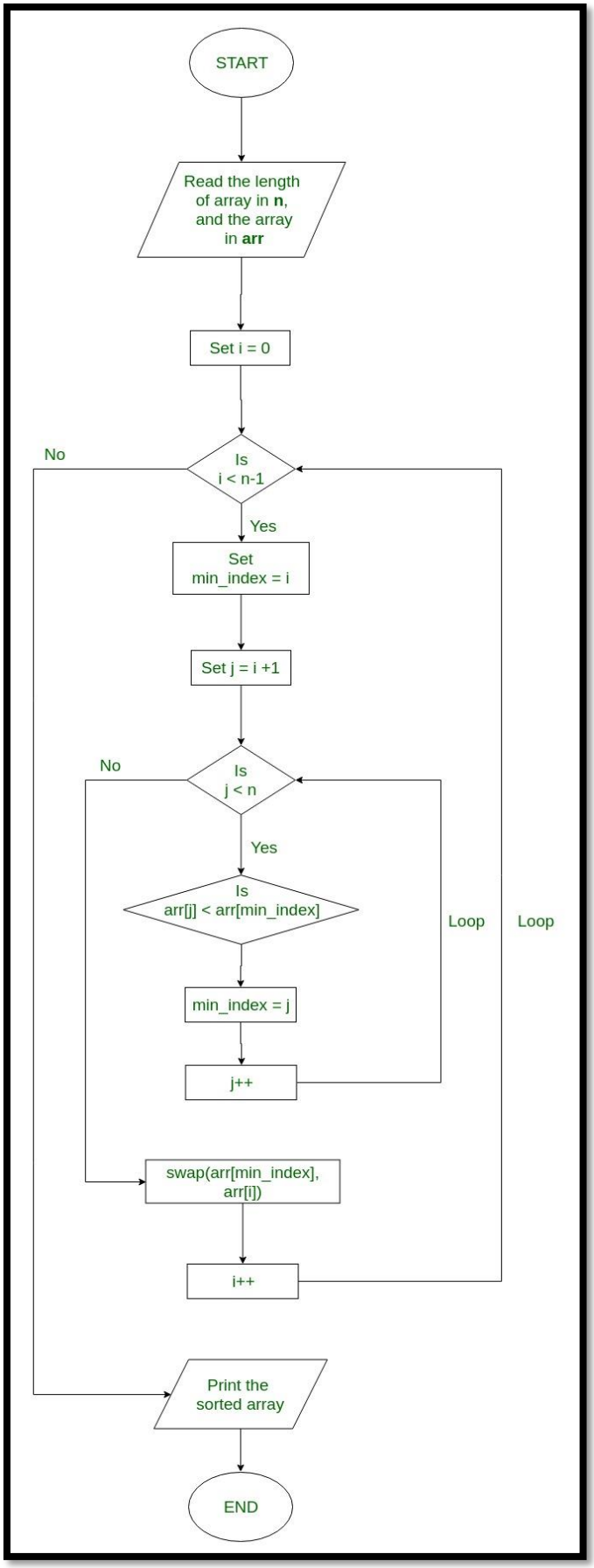
En cada iteración de este algoritmo, el elemento mínimo del sub arreglo no ordenado es seleccionado y mandado al sub arreglo ordenado.

- *EJEMPLO*

```
arr[] = 64 25 12 22 11

// Find the minimum element in arr[0..4] and place it at
beginning

11 25 12 22 64

// Find the minimum element in arr[1...4] and place it at
beginning of arr[1...4]
11 12 25 22 64

// Find the minimum element in arr[2...4] place it at beginning
of arr[2...4]
11 12 22 25 64

// Find the minimum element in arr[3...4] and place it at
beginning of arr[3...4]
11 12 22 25 64
```

- *PSEUDOCÓDIGO Y DIAGRAMA DE FLUJO*

```
para i=0 hasta n-1
    para j=i+1 hasta n
        si lista[j] < lista[i] entonces
            intercambiar(lista[i], lista[j])
        fin si
    fin para
fin para
```


START
Read the length of array in n, and the array in arr
Set i = 0
No
Is i < n-1
Yes
Set min_index = i
Set j = i +1
No
Is j < n
Yes
Is arr[j] < arr[min_index]
Loop
Loop
min_index = j
j++
swap(arr[min_index], arr[i])
i++
Print the sorted array
END

- *ANÁLISIS*
  - ✓ El bucle exterior se realiza n-1 veces
  - ✓ El bucle interior se ejecuta n/2 veces en promedio
  - ✓ El trabajo realizado dentro del bucle interno es constante
  - ✓ El tiempo requerido es aproximadamente: $(n-1) * (\frac{n}{2})$
  - ✓ El tiempo de complejidad por lo tanto es: $O\ (n^2)$
  - ✓ El espacio auxiliar es de: O (1), no requiere un espacio extra.

- *CÓDIGO EN C++*

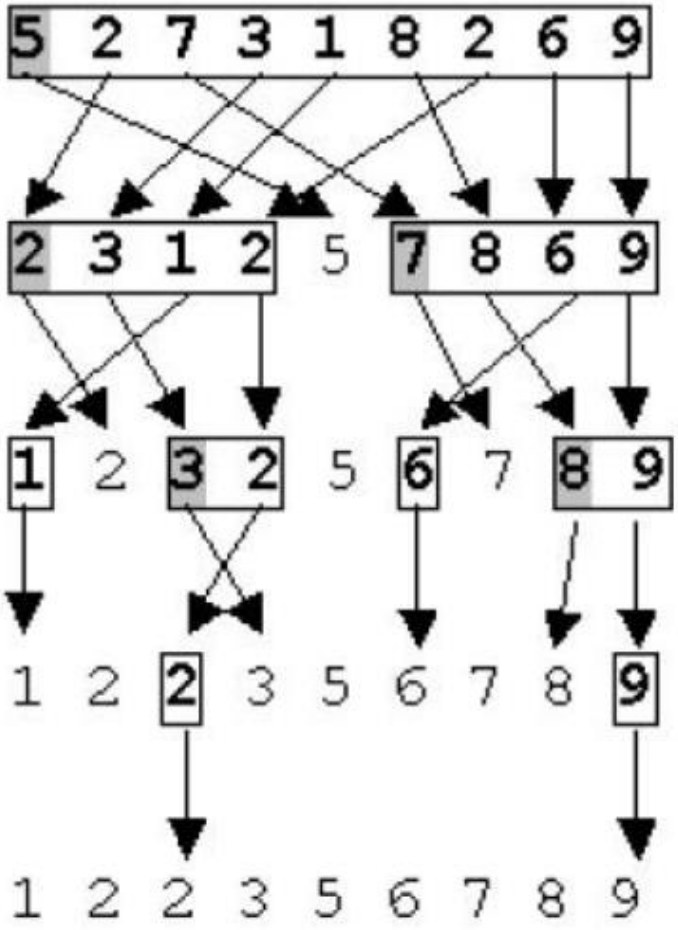
```
#include <iostream>
using namespace std;

void ordena(int a[5])
{for(int i=0;i<5;i++)
    {for (int j=i;j<5;j++)
      {if (a[i] > a[j])
         {int aux = a[j];
        a[j] = a[i];
        a[i] = aux; }}}
}

int main()
{int a[5];
 for (int i=0;i<5;i++)
  {cout << "Ingrese el " << i <<"° elemento" << endl;
   cin >> a[i];
   cout << "\n\n";
 }
 ordena(a);
 cout << "Lista ordenada" << endl;
 for(int i=0; i < 5; i++)
   {cout << a[i] << endl; }
 return 1;
}
```

2. Quick Sort

Como segundo algoritmo de ordenamiento tenemos al Quick Sort que tiene cierto parecido con Merge Sort debido a que ambos usan la técnica de "divide y vencerás". Escoge un elemento como pivote y divide el arreglo dado con respecto al pivote seleccionado: en un lado los elementos menores al pivote; por otro lado los elementos mayores al pivote) . La lista queda separada en dos sublistas, después se debe de repetir este proceso de forma recursiva para cada sublista mientras éstas contengan más de un elemento. Una vez terminado este proceso todos los elementos estarán ordenados.

Debido que en este método se debe seleccionar un elemento del arreglo como pivote, existen distintas versiones de este ordenamiento como son: Elegir el primer elemento como pivote, elegir el último como pivote o elegir el pivote de manera aleatoria.

- *EJEMPLO*



- *PSEUDOCÓDIGO*

```
ACCION QuickSort (A,inf,sup)
     ENTERO i,j
     i<-inf
     j<-sup
     x<-A[(inf+sup)/2]
     MIENTRAS (i<=j)
          MIENTRAS (A[i]<x)
               i <- i+1
          FIN_MIENTRAS
          MIENTRAS (A[j]>x)
               j <- j-1
          FIN_MIENTRAS
          SI (i<=j)
```

```
                tam <- A[i]
                A[i] <- A[j]
                A[j] <- tam
                i <- i+1
                j <- j-1
          FIN_SI
     FIN_MIENTRAS
     SI (inf<j)
          QuickSort (A,inf,j)
     FIN_SI
     SI (i<sup)
          QuickSort (A,i,sup)
     FIN_SI
FIN_ACCION
```

- *ANÁLISIS*
  - ✓ Tiempo de ejecución es el siguiente: $T(n) = T(k) + T(n-k-1) + \Theta(n)$, donde k es el numero de elementos menores al pivote.
  - ✓ Ordena en tiempo θ (n^2) en el peor caso (cuando se escoge el mayor o menor elemento del array como pivote)
  - ✓ Sin embargo, su tiempo promedio es de O(n*logn)

- *CÓDIGO EN C++*

```
#include <bits/stdc++.h>
using namespace std;

int partition (int arr[], int low, int high) {
   int pivot = arr[high]; // pivot
   int i = (low - 1); // Index of smaller element

   for (int j = low; j <= high - 1; j++) {
         if (arr[j] < pivot) {
               i++; // increment index of smaller element
               swap(&arr[i], &arr[j]);
         }
   }
   swap(&arr[i + 1], &arr[high]);
   return (i + 1);
}

void quickSort(int arr[], int low, int high) {
   if (low < high) {
         int pi = partition(arr, low, high);
         quickSort(arr, low, pi - 1);
         quickSort(arr, pi + 1, high);
   }
}

void printArray(int arr[], int size) {
```

```
    int i;
    for (i = 0; i < size; i++)
          cout << arr[i] << " ";
    cout << endl;
}

int main() {
    int arr[] = {10, 7, 8, 9, 1, 5};
    int n = sizeof(arr) / sizeof(arr[0]);
    quickSort(arr, 0, n - 1);
    cout << "Sorted array: \n";
    printArray(arr, n);
    return 0;
}
```

3. Counting Sort

Por último, tenemos el Counting Sort que es un algoritmo de ordenamiento basada en números de un rango específico. Su funcionamiento comienza a través del conteo del número de objetos que tengan valores de clave distintos. Luego, se realizan cálculos de aritmética para calcular la posición de cada objeto en la secuencia de salida.

- *EJEMPLO*

```
For simplicity, consider the data in the range 0 to 9.
Input data: 1, 4, 1, 2, 7, 5, 2
  1) Take a count array to store the count of each unique.
  Index:     0  1  2  3  4  5  6  7  8  9
  Count:     0  2  2  0   1  1  0  1  0  0

  2) Modify the count array such that each element at each index
  stores the sum of previous counts.
  Index:     0  1  2  3  4  5  6  7  8  9
  Count:     0  2  4  4  5  6  6  7  7  7

The modified count array indicates the position of each object
in the output sequence.

  3) Output each object from the input sequence followed by
  decreasing its count by 1.
  Process the input data: 1,4,1,2,7,5,2. Position of 1 is 2.
```

Put data 1 at index 2 in output. Decrease count by 1 to place next data 1 at an index 1 smaller than this index.

- *PSEUDOCÓDIGO*

```
ACCION CountingSort (A,B,k)
      ENTERO i,j
      PARA i<-1 HASTA k
            C[i] <- 0
      FIN_PARA
      PARA j<-1 HASTA n
            C[A[j]] <- C[A[j]] + 1
      FIN_PARA
      PARA i<-2 HASTA k
            C[i] <- C[i] + C[i-1]
      FIN_PARA
      PARA j<-n HASTA 1
            B[C[A[j]]] <- A[j]
            C[A[j]] <- C[A[j]] - 1
      FIN_PARA
FIN_ACCION
```

- *ANÁLISIS*
  - ✓ No realiza comparaciones entre elementos
  - ✓ Tiempo de complejidad → O(n+k) donde k es el rango del input
  - ✓ Espacio auxiliar → O(n+k)
  - ✓ También puede ser utilizado para números negativos

- *CÓDIGO EN C++*

```
#include <algorithm>
#include <iostream>
#include <vector>
using namespace std;

void countSort(vector<int>& arr) {
   int max = *max_element(arr.begin(), arr.end());
```

```
    int min = *min_element(arr.begin(), arr.end());
    int range = max - min + 1;

    vector<int> count(range), output(arr.size());
    for (int i = 0; i < arr.size(); i++)
        count[arr[i] - min]++;

    for (int i = 1; i < count.size(); i++)
        count[i] += count[i - 1];

    for (int i = arr.size() - 1; i >= 0; i--) {
        output[count[arr[i] - min] - 1] = arr[i];
        count[arr[i] - min]--;
    }

    for (int i = 0; i < arr.size(); i++)
        arr[i] = output[i];
}

void printArray(vector<int>& arr) {
    for (int i = 0; i < arr.size(); i++)
        cout << arr[i] << " ";
    cout << "\n";
}

int main() {
    vector<int> arr = { -5, -10, 0, -3, 8, 5, -1, 10 };
    countSort(arr);
    printArray(arr);
    return 0;
}
```

Luego de hablar de los algoritmos de ordenamiento, hablaremos sobre las herramientas que usamos para el proyecto de desarrollo de la aplicación web.

JAVA SCRIPT

¿Qué es JavaScript?

JavaScript es un lenguaje de programación de scripts (secuencia de comandos) orientado a objetos por ende se define los siguientes términos.

En primer lugar, un lenguaje de programación es un lenguaje que permite a los desarrolladores escribir código fuente que será analizado por un ordenador.

Un desarrollador o programador es una persona que desarrolla programas. Puede ser un profesional (un ingeniero, programador informático o analista) o un aficionado.

El código fuente está escrito por el desarrollador. Este es un conjunto de acciones, llamadas instrucciones, lo que permitirá dar órdenes al ordenador para operar el programa.

El código fuente es algo oculto, como un motor en un automóvil está oculto, pero está ahí, y es quien asegura que el coche puede ser conducido. En el caso de un programa, es lo mismo, el código fuente rige el funcionamiento del programa.

Dependiendo del código fuente, el ordenador realiza varias acciones, como abrir un menú, iniciar una aplicación, efectuar búsquedas, en fin, todo lo que el equipo es capaz de hacer.

- Scripts de programación

JavaScript permite programar scripts. Este es un lenguaje de programación que será utilizado para escribir código fuente para ser analizada por un ordenador. Hay tres formas en que se usa el código fuente:

  - ✓ Lenguaje compilado: el código fuente se da a un programa llamado compilador que lee el código fuente y lo convierte en un lenguaje que el equipo será capaz de interpretar: el lenguaje binario, es de 0 y 1. Lenguajes como C o C ++ son lenguajes compilados muy conocidos.
  - ✓ Lenguaje precompilado: el código fuente se compila en parte, por lo general en un código más fácil de leer para el ordenador, pero que todavía no es binario. Este código intermedio es para ser leído por lo que se llama una “Máquina Virtual", que ejecutará el código. Lenguajes como C # o Java se llaman precompilados.
  - ✓ Lenguaje interpretado: no hay compilación, el código fuente se mantiene sin cambios, y si desea ejecutar este código, debemos proporcionar un intérprete que va a leer y realizar las acciones solicitadas

- Lenguaje orientado a objetos

Queda un aspecto a analizar: orientado a objetos. Un lenguaje de programación orientado a objetos es un lenguaje que contiene elementos, llamados objetos y los objetos diferentes tienen características específicas y formas de uso diferente. El lenguaje proporciona objetos básicos, como imágenes, fechas, cadenas de caracteres, etc. También es posible crear tus propios objetos para hacer la vida más fácil y obtener un código fuente más claro y una forma de programar mucho más intuitivo.

JavaScript, el lenguaje de scripts

JavaScript actualmente es principalmente utilizado en internet, junto con las páginas web (HTML o XHTML). JavaScript está directamente incluido en la página web y mejora una página HTML, añadiendo interacción del usuario, animación, ayudas a la navegación, tales como:

- Mostrar / ocultar el texto
- Deslizamiento de imágenes
- Crear presentaciones de diapositivas
- Crear burbujas de información

Esto es importante porque el propósito de los scripts del lado del cliente y del lado del servidor no es el mismo. Un script del lado del servidor se encargará de "crear" la página web que se envía al navegador.

JavaScript no es la Web

Si JavaScript está diseñado para ser usado en conjunción con HTML, el lenguaje ha evolucionado desde entonces hacia otros destinos. JavaScript es regularmente utilizado para hacer extensiones para diferentes programas, como los scripts codificados en Lua o Python.

JavaScript también se puede utilizar para construir aplicaciones. Mozilla Firefox es el ejemplo más famoso: la interfaz del navegador se crea con una especie de HTML llamado XUL y JavaScript que se utiliza para animar la interfaz. Otros programas también están basados en esta tecnología, como por ejemplo de TomTom HOME que se utiliza para administrar el navegador GPS TomTom a través de la PC.

Breve historia del lenguaje

En 1995, Brendan Eich trabajaba en Netscape Communications Corporation, la compañía que publicó el famoso Netscape Navigator, entonces principal competidor de Internet Explorer. Brendan desarrolló Live Script un lenguaje de script que se basa en el lenguaje Java, y que estaba destinado a ser instalado en los servidores desarrollados por Netscape. Netscape inició el desarrollo de una versión del cliente LiveScript, que pasó a llamarse JavaScript, en homenaje al lenguaje Java creado por Sun Microsystems. En efecto, en ese momento, el lenguaje Java era cada vez más popular, y Brendan Eich, el padre de JavaScript, al llamarlo JavaScript en lugar de LiveScript era una forma de publicidad de

Java y JavaScript en sí. Al final, estos dos lenguajes son radicalmente diferentes, no se vaya a confundir Java y JavaScript, pues operan de forma diferente.

JavaScript fue lanzado en diciembre de 1995 y estaba integrado en Netscape Navigator 2. El lenguaje fue tan exitoso, por lo que Microsoft desarrolló una versión similar llamada JScript, que se instalaba en Internet Explorer 3, en 1996. Netscape decidió enviar a su versión de JavaScript a ECMA International (European Computer Manufacturers Asociación, la Asociación Europea de Normalización de hoy los sistemas de información y comunicación) para que el lenguaje fuera normalizado, es decir para que se creara una referencia del lenguaje y que así pudiera ser utilizado por otras personas y embebidos en otro software. ECMA International estandarizó el lenguaje con el nombre de ECMAScript. Desde entonces, las versiones de ECMAScript han evolucionado. La versión más conocida es utilizada en todo el mundo, es la versión ECMAScript 3, publicado en diciembre de 1999.

Cascading Style Sheets

¿Qué es CSS?

CSS (en inglés Cascading Style Sheets) es lo que se denomina lenguaje de hojas de estilo en cascada y se usa para estilizar elementos escritos en un lenguaje de marcado como HTML. CSS separa el contenido de la representación visual del sitio. CSS fue desarrollado por W3C (World Wide Web Consortium) en 1996 por una razón muy sencilla. HTML no fue diseñado para tener etiquetas que ayuden a formatear la página. Está hecho solo para escribir el marcado para el sitio. Se incluyeron etiquetas como <font> en HTML versión 3.2, y esto les causó muchos problemas a los desarrolladores. Dado que los sitios web tenían diferentes fuentes, fondos de colores y estilos, el proceso de reescribir el código fue largo, doloroso y costoso. Por lo tanto, CSS fue creado por W3C para resolver este problema.

La relación entre HTML y CSS es muy fuerte. Dado que HTML es un lenguaje de marcado (es decir, constituye la base de un sitio) y CSS enfatiza el estilo (toda la parte estética de un sitio web).CSS no es técnicamente una necesidad, pero es preferible tener un sitio que tenga HTML y CSS.

Ventajas de CSS

La diferencia entre un sitio web que implementa CSS y uno que no, es enorme. Cuando algún sitio web que no se puede cargar por completo y tiene un fondo blanco con la mayor parte del texto en azul y negro (Así es como se ve un sitio con solo HTML). Eso significa que la parte CSS del sitio no se cargó correctamente o no existe.

Antes de CSS, todo el estilo debía incluirse en el marcado HTML. Esto significa que había que describir por separado todos los fondos, los colores de fuente, las alineaciones, etc. CSS permite estilizar todo en un archivo diferente, creando el estilo allí y después integrando el archivo CSS sobre el marcado HTML. Esto hace que el marcado HTML sea mucho más limpio y fácil de mantener. En resumen, con CSS no tienes que describir repetidamente cómo se ven los elementos individuales. Esto ahorra tiempo, hace el código más corto y menos propenso a errores.

CSS te permite tener múltiples estilos en una página HTML, y esto hace que las posibilidades de personalización sean casi infinitas. Hoy en día, esto se está volviendo una necesidad más que algo básico.

Cómo funciona CSS

CSS utiliza una sintaxis simple basada en el inglés con un conjunto de reglas que la gobiernan. HTML no fue hecho con la intención de utilizar elementos de estilo, sino solo para el marcado de la página. Fue creado simplemente para describir el contenido. Por ejemplo: "* Esto es un párrafo. *".

La estructura de sintaxis CSS es bastante simple. Cuenta con un selector y un bloque de declaración. Primero se selecciona un elemento y luego declaras lo que se hará con él. Sin embargo, hay reglas que se debe recordar. El selector apunta al elemento HTML que se estilizara. El bloque de declaración contiene una o más declaraciones separadas por punto y coma. Cada declaración incluye un nombre de propiedad CSS y un valor, separados por dos puntos. Una declaración CSS siempre termina con un punto y coma, y los bloques de declaración están rodeados por llaves.

Estilos de implementación de CSS

Los diferentes estilos de CSS son Inline, Externo e Interno.

- Estilo Interno. Los estilos CSS hechos de esta manera se cargan cada vez que se actualiza un sitio web, lo que puede aumentar el tiempo de carga. Además, no se podrá usar el mismo estilo CSS en varias páginas, ya que está contenido en una

sola página. Sin embargo, tener todo en una página facilita compartir la plantilla para una vista previa.

- Estilo Externo. Todo se hace externamente en un archivo "*.css*". Esto significa que se puede hacer todo el estilizado en un archivo separado y aplicar el CSS a cualquier página además también puede mejorar los tiempos de carga.
- Estilo Inline. Trabaja con elementos específicos que tienen la etiqueta <style>. Cada componente tiene que ser estilizado, su función principal es cambiar un solo elemento, tener una vista previa rápida de los cambios o cuando no se tenga acceso a los archivos CSS.

HyperText Markup Language (HTML)

¿Qué es HTML?

HTML es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet. Se trata de la sigla que corresponde a HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto, que podría ser traducido como Lenguaje de Formato de Documentos para Hipertexto.

Se trata de un formato abierto que surgió a partir de las etiquetas SGML (Standard Generalized Markup Language). Concepto traducido generalmente como «Estándar de Lenguaje de Marcado Generalizado» y que se entiende como un sistema que permite ordenar y etiquetar diversos documentos dentro de una lista. Este lenguaje es el que se utiliza para especificar los nombres de las etiquetas que se utilizarán al ordenar, no existen reglas para dicha organización, por eso se dice que es un sistema de formato abierto.

EL HTML se encarga de desarrollar una descripción sobre los contenidos que aparecen como textos y sobre su estructura, complementando dicho texto con diversos objetos (como fotografías, animaciones, etc.).

Es un lenguaje muy simple y general que sirve para definir otros lenguajes que tienen que ver con el formato de los documentos. El texto en él se crea a partir de etiquetas, también llamadas tags, que permiten interconectar diversos conceptos y formatos.

Para la escritura de este lenguaje, se crean etiquetas que aparecen especificadas a través de corchetes o paréntesis angulares: < y >. Entre sus componentes, los elementos dan

forma a la estructura esencial del lenguaje, ya que tienen dos propiedades (el contenido en sí mismo y sus atributos).

Por otra parte, cabe destacar que el HTML permite ciertos códigos que se conocen como scripts, los cuales brindan instrucciones específicas a los navegadores que se encargan de procesar el lenguaje. Entre los scripts que pueden agregarse, los más conocidos y utilizados son JavaScript y PHP.

El marcado estructural es el que estipula la finalidad del texto, aunque no define cómo se verá el elemento. El marcado presentacional, por su parte, es el que se encarga de señalar cómo se verá el texto más allá de su función.

Para conocer el código HTML que utiliza una página web, hay que seleccionar Ver código fuente en nuestro navegador (como Internet Explorer o Mozilla Firefox). Al elegir esta opción, se abrirá el editor de texto con el código HTML de la página que se está visualizando.

Breve historia del HTML

Este lenguaje fue desarrollado por la Organización Europea de Investigación Nuclear (CERN) en el año 1945 con la finalidad de desarrollar un sistema de almacenamiento donde las cosas no se perdieran, que pudieran ser conectadas a través de hipervínculos. Primeramente, crearon un dispositivo llamado «memex», el cual era considerado como un suplemento para la memoria. Posteriormente, Douglas Engelbart, diseñó un entorno de trabajo por computadora que recibiría el nombre de oNLine System que poseía un catálogo para facilitar la tarea de búsqueda dentro de un mismo organismo.

Recién en 1965, Ted Nelson acuñó el término hipervínculo, ideando una estructura que se encontraba conectada de forma electrónica y que más tarde permitiría la creación de la World Wide Web (1989), un sistema de hipertexto a través del cual era posible compartir una variada información sirviéndose de Internet (servía para la comunicación entre investigadores nucleares que formaran parte del CERN). El norteamericano Tim Berners-Lee fue el primero en proponer una descripción de HTML en un documento que publicó en 1991. Allí describía veintidós componentes que suponen el diseño más básico y simple del HTML.

El tipo de codificación que se utilizó para el desarrollo de este sistema de hipervínculos debía ser comprendido, tanto por ordenadores tontos como por mega-estaciones, por eso

fue necesario crear uno absolutamente simple, tanto en lo que respectaba al lenguaje de intercambio (HTML), como el que hacía referencia al protocolo de red (HTTP).

Al día de hoy existen los Editores Web que permiten que los diseñadores, a través de herramientas gráficas que reciben el nombre de WYSIWYG puedan crear páginas web sin conocer el código HTML, este se crea de forma automatizada, dándole estructura a la web y permitiendo que sea más allá del ordenador donde es creada. Entre los recursos que pueden enlazarse al código HTML se encuentran fotografías, vídeos, archivos de otras webs o incluso de la misma y todo tipo de contenido que se encuentre subido a la red.

Por último, se describirá de manera concisa la librería que se usó que fue vue:

LIBERÍA VUE

Vue es un framework open source de JavaScript, el cual nos permite construir interfaces de usuarios de una forma muy sencilla.

Vue fue creado por Evan You ex trabajador de Google, quien, es importante mencionar, fue desarrollador Angular. Vue fue lanzado en el año 2014. Aunque inicialmente fue pensado para ser una biblioteca personal, la comunidad hizo que el proyecto creciera a un ritmo impresionante, posicionándolo hoy en día como uno de los Frameworks web más populares, junto con Angular y React. Una de las características más importantes de Vue es el trabajo con componentes. Un componente Vue, en términos simples, es un elemento el cual se encapsula código reutilizable. Dentro de un componente podremos encontrar etiquetas HTML, estilos de CSS y código JavaScript.

Características

Tal como comentábamos VueJS es un framework pensado para hacer las cosas más simples. Pero si todavía tienes dudas para “adentrarte en sus dominios” quizá un repaso por sus características generales te ayude a decidirte.

1. Un framework progresivo. Podemos incluir las partes que necesitamos además de la librería core y luego incluir otras librerías como si fueran módulos separados. Esto permite añadir funcionalidad a medida que lo necesitemos. El core principal de VueJS está formado por una librería encargada de renderizar vistas en el navegador.

2. Intuitivo, moderno y fácil de usar. VueJS no ha reinventado la rueda pero desde sus comienzos su premisa es que sea tan fácil y simple de usar que la comunidad pueda usarlo para sus proyectos sin apenas curva de aprendizaje.
3. Un ecosistema muy variado que cubre todo lo necesario. VueJS tiene a su alrededor una serie de herramientas que ayudan a conseguir que el desarrollador sepa en todo momento qué está haciendo y cómo lo está haciendo. Desde un interface de línea de comandos (CLI) a una plantilla base que podemos invocar mediante la herramienta vue-cli disponible en los repositorios npm, pasando por un largo etcétera de módulos y complementos.
4. Una comunidad muy activa. Revisiones constantes, buena documentación, comunidad dispuesta a echar una mano cuando estás atascado. Un lujo.
5. Todo el código de un componente se encuentra en un único fichero. Los componentes guardan todo lo necesario en ficheros con extensión .vue. Estos ficheros contienen de manera ordenada todo el HTML, el CSS y el JavaScript necesario.

Si queremos hacer uso de Vue no será necesario instalar absolutamente nada en nuestra computadora, si así lo deseamos podemos utilizar este framework mediante un CDN.

La forma más fácil de probar Vue.js es usando el ejemplo Hola Mundo en JSFiddle. Siéntase libre de abrirlo en otra pestaña y seguirlo mientras analizamos algunos ejemplos básicos. O puede crear un archivo index.html e incluir Vue con alguna de estas opciones:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.4.2/vue.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<script src="https://cdn.jsdelivr.net/npm/vue"></script>
```

Ejemplo

Veamos un ejemplo

```
<div id="app">
  <div class=books v-for"book in books"
   <p> Titulo : {{ book.title }} </p>
    <div  v-if="book.has_description">
      <p> {{ book.description }} </p>
    </div>
    <div v-else>
      <p> El libro no cuenta con una descripción! </p>
```

```
    </div>
  </div>
</div>
```

En este ejemplo podemos observar cómo iteramos una colección de libros, en cada iteración pintamos el título. Si el libro tiene descripción entonces la mostramos, en caso contrario mostramos el mensaje, "¡El libro no cuenta con una descripción!".

METODOLÓGIA DE DESARROLLO

Se agregó variables que almacenarán los valores de:

Para crear la lista inicial:

- Entero que indica el tamaño de la lista inicial.
- Booleano que indica si se quiere agregar valores aleatoriamente a la lista inicial
- Entero que indica una posición en la lista inicial en caso se desee agregar un valor en la lista inicial.
- Entero que indica un valor que se desee agregar en la lista inicial.
- Lista de enteros que representa la lista inicial de elementos a ordenar.
- Booleano que indica si desea realizar el ordenamiento con animación.
- Entero que indica la velocidad con la cual se ejecuta la animación.
- Valor que almacena el tipo de ordenamiento que se realizará.

Para ordenar la lista inicial:

- Valor que almacena el nombre del ordenamiento seleccionado.
- Valor que almacena el tiempo transcurrido cuando se ordena con animación.
- Lista de enteros que representa la lista que se ordenará.
- Lista de enteros en donde se ordenará la lista inicial.

Para la totalidad del proyecto se usa HTML y CSS para la interfaz y para la parte de la lógica se usa JavaScript nativo y vue.js.

La primera parte del proyecto se centra en la creación de la lista de elementos que se ordenará, se podrá decidir el tamaño de la lista, así como agregar los elementos que uno

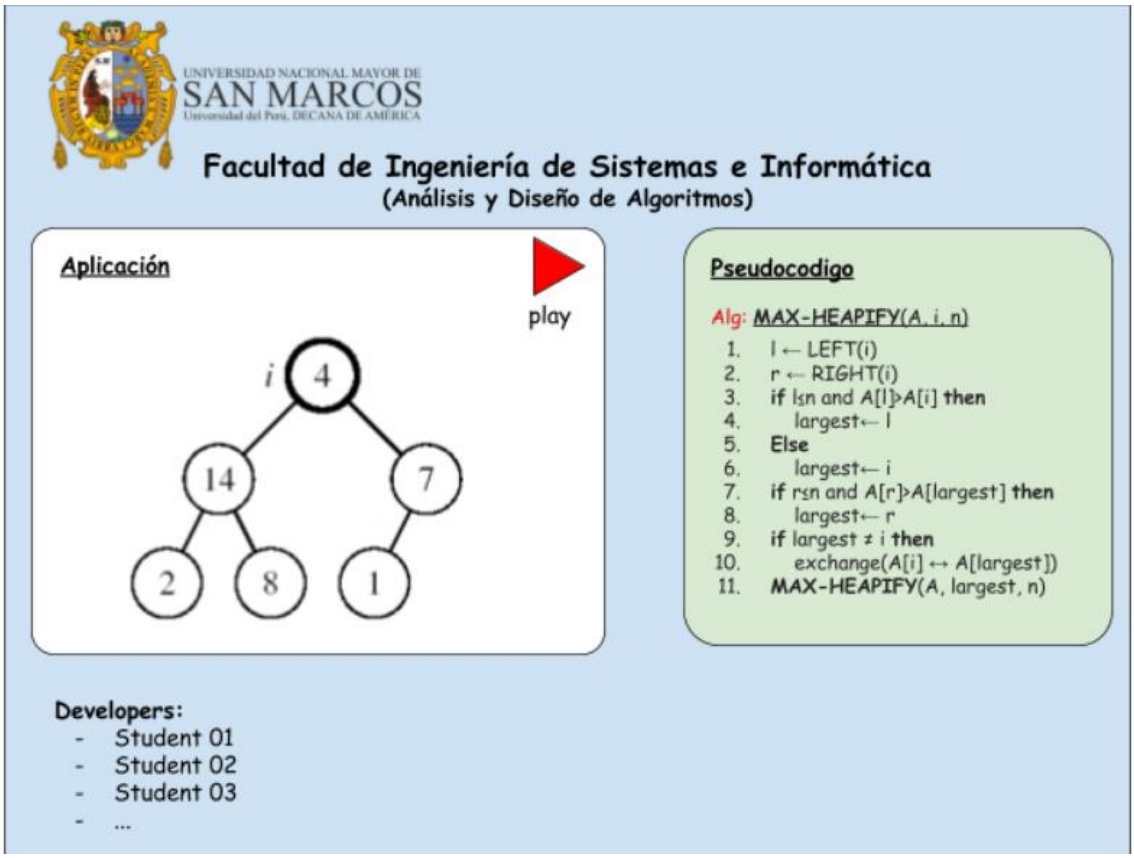
decida en el índice que uno desee, también podrá elegir el tipo de ordenamiento y la opción de ordenarlo con animación o sin ella.

La segunda parte del proyecto se centra en la lógica del ordenamiento de la lista de elementos dependiendo del ordenamiento seleccionado, el ordenamiento con animación para sus efectos se realiza con pausas de tiempos en cada paso del algoritmo de ordenamiento, estas pausas fueron puestas en determinados momentos para que se pueda entender la lógica de cada ordenamiento.

Vue.js se usa para poder atar los elementos definidos en js mencionados anteriormente a la interfaz, o sea el HTML, esto hace posible los cambios de valores de cada uno de los elementos definidos en JavaScript se refleje en tiempo real en la parte visual, esto se usa más que nada para el tema de la animación, así cada elemento modificado en la lista de elementos al momento de la ordenación se verá reflejado en la interfaz lo que dará el efecto de animación.

Para el ordenamiento de Counting Sort se usará dos listas adicionales que se usarán como ayudantes, pero trabajan bajo el mismo concepto, son listas que reflejan sus valores en tiempo real con determinadas pausas para que puedan ser apreciadas.

En este proyecto no hemos usado clases por lo que no hemos realizado un diagrama de este ni tampoco un diagrama de secuencia; sin embargo, hemos utilizado unas plantillas como inspiración para la interfaz. Las cuales fueron sacadas de la especificación del proyecto y se mostrara a continuación:



CONCLUSIONES Y RECOMENDACIONES

Luego de realizar este proyecto, dejamos las siguientes conclusiones y recomendaciones:

- Es importante aprender los diversos algoritmos ya que estos son de suma importancia en el momento de construir programas en el futuro.
- No solo basta con memorizarlos, sino también emplearlos en las tareas o proyectos futuros. Ya que esto permitirá que los usemos con más facilidad y que adoptemos un nivel de lógica mayor a la hora de resolver problemas.
- Existen algunos algoritmos que a la hora de implementarlos tuvimos que poner “pausas” para que se aprecie mejor la dinámica y se entienda el proceso que realiza el algoritmo. En nuestro caso lo hicimos con los 3 ordenamientos.
- Herramientas como HTML y CSS resultaron muy útiles para el diseño de la aplicación web.
- Otras herramientas como GitHub, fueron beneficiosas para la gestión de los avances del código.

REFERENCIAS

1. B., G. (13 de mayo de 2019). *Hostinger Tutoriales*. Obtenido de https://www.hostinger.es/tutoriales/que-es-css/
2. GeekforGeeks. (2 de Mayo de 2019). *GeekforGeeks*. Obtenido de https://www.geeksforgeeks.org/selection-sort/
3. GeekforGeeks. (4 de Setiembre de 2020). *GeekforGeeks*. Obtenido de https://www.geeksforgeeks.org/quick-sort/
4. GeekforGeeks. (2 de Setiembre de 2020). *GeekforGeeks*. Obtenido de GeekforGeeks: https://www.geeksforgeeks.org/counting-sort/
5. Menendez, R., & Barzanallana, A. (2015). *JavaScript.* Madrid.
6. Perez Porto, J., & Gardey, A. (2012). Obtenido de https://definicion.de/html/
7. Pino, J. J. (Diciembre de 2019). *DevCode*. Obtenido de https://devcode.la/blog/que-es-html/
8. https://codigofacilito.com/articulos/que-es-vue
9. https://ifgeekthen.everis.com/es/vuejs-el-framework-javascript-que-te-hace-la-vida-mas-facil
10. Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2007). Introduction to algorithms (3nd ed.). MIT Press.
11. https://www.youtube.com/watch?v=seaq4UxKNHU