ServiApp

Felipe Rojas, Fabian López, Victor Barragán

No. grupo del curso: {*4*} y No. de Equipo de Trabajo: {*4*}

I. INTRODUCCIÓN

Este documento consiste en la presentación formal del proyecto en su 1ª entrega para la materia 'Estructuras de datos'. El proyecto consiste en el desarrollo de una página web denominada "ServiApp", se presentarán a continuación los avances en interfaz gráfica, software además de distintas pruebas de datos, funcionalidades, dificultades en el desarrollo, entre otros.

I. DESCRIPCIÓN DEL PROBLEMA

Existen distintas aplicaciones que se encargan de reunir servicios y sus proveedores con el fin de facilitar este proceso a los usuarios, en su mayoría son aplicaciones de domicilios enfocadas a la gastronomía y aplicaciones de transporte.
Sin embargo, no existe una aplicación que ofrezca servicios técnicos, especializados o poco frecuentes. Además, de unificar todos estos servicios en una aplicación para comodidad del usuario.
Otro aspecto para considerar es la valoración de los servicios ofrecidos, esta valoración promueve y permite brindar la mejor experiencia a los usuarios además de fortalecer la sana competencia de los proveedores, todo esto se puede realizar gracias al avance de la tecnología y resulta poco provechoso que los servicios y productos ofrecidos tradicionalmente aun no cuenten con la virtualización necesaria para aprovechar las tendencias tecnológicas.
Con lo anterior, ServiApp tiene como objetivo ofrecer servicios técnicos (plomería, eléctricos, entre otros) resaltando una contratación justa y libre para ambos perfiles (usuarios y proveedores) en donde se estipulen los términos entre ambas partes antes de que se realice el servicio.

II. USUARIOS DEL PRODUCTO DE SOFTWARE

ServiApp cuenta con 3 tipos de roles diferentes en cuanto a sus usuarios se refiere, estos son:

- Usuario informal: Es aquel usuario que ingresa a la página web sin una verificación de su cuenta (sin inicio de sesión), este usuario puede acceder a la información de contacto suministrada por los distintos proveedores de servicios y realizar una contratación por su cuenta.
- Usuario Formal: Este usuario posee y utiliza su cuenta en la aplicación para realizar contrataciones de un servicio y revisar en su perfil contrataciones pasadas.
- Usuario Proveedor: Son aquellos usuarios que desean registrarse en los distintos servicios como proveedores, deben suministrar información pertinente para la realización de los contratos y un posible contacto con los usuarios que deseen contratar sus servicios.

III. REQUERIMIENTOS FUNCIONALES DEL SOFTWARE

Para la lectura de los requerimientos funcionales de este software hay que tener en cuenta que la definición de "página" no necesariamente refiere a la totalidad de la página como tal, también puede referir a una sección de la misma aludiendo a la forma como se ejecuta el código en cada sección y a la forma como funcionan las conexiones entre las distintas secciones a partir de "hipertextos" o más comúnmente conocidos como "enlaces" que en términos simples son las conexiones existentes entre elementos de la página (textos, botones, imágenes, etc) que redirigen al usuario ya sea a otra sección de la misma página web (refiérase a página web como a la totalidad del software) o a otra página web distinta a "ServiApp".

Registro de usuarios (Formales y proveedores)

- *Descripción:* Permite a un usuario informal (visitante de la página web) registrar sus datos en la base de datos de la página web de modo que su información quede guardada para poder hacer uso de las distintas funcionalidades de ServiApp que requieran del continuo almacenamiento de estos.

- *Acciones iniciadoras y comportamiento esperado*:

1. Luego del acceso por parte del usuario en la página, este deberá hacer clic al botón de "registrarse".
2.
3. La página se redireccionará a otra dónde se encontrará un formulario el cual deberá diligenciar para su registro.
4. Una vez el usuario de clic en el botón ubicado al final del formulario se desencadenará un evento de tipo "submit" el cuál enviará los datos diligenciados en el formulario como un objeto (clase Usuario o clase Proveedor respectivamente) al código fuente de la página.
5. A partir de aquí se ejecutará un método en donde el objeto creado se enviará y almacenará en una base de datos.
6. (Adicionalmente el usuario antes de ingresar a este formulario deberá haber hecho click en el botón "iniciar sesión" de modo que el registro que se haga quede registrado en la base de datos con un correo electrónico, esto debido a la forma como actualmente se está implementando el API)

Requerimientos funcionales:

- Disponibilidad del enlace al formulario para los usuarios informales que visiten la página web.
    - Esto básicamente se resume en la disponibilidad de la página que contiene el formulario y que tanto esta como el enlace que redirecciona aquí no contengan errores.
- Conexión activa entre los archivos que contienen el código fuente de la página y los usados para la conexión con la base de datos (back-end).
    - Cabe agregar que para cada entrada del formulario se especifica un tipo de dato especifico y unas longitudes específicas que en dado de no cumplirse por parte del usuario se le indicará bajo la figura de un aviso en pantalla e imposibilitará él envió del formulario al deshabilitar el evento "submit" hasta que se corrijan los datos ingresados.
    - Si esta conexión no es posible, existirá una excepción para el posible error que ejecutará un aviso en pantalla que informará al usuario sobre el estado de la página web. Esto posiblemente desencadenará un aviso que le será enviado a los desarrolladores para corregir el error lo más pronto posible.
- Disponibilidad de la base de datos para recibir las peticiones por y ejecutar los métodos para almacenar estos datos.
    - Esto se resume en el hecho de que exista la conexión estable y vigente con la base de datos y que esta esté funcionando plenamente. Para ello se comprueba constantemente que los datos se envíen y se guarden adecuadamente.

Eliminación de usuarios (Formales y proveedores)

- *Descripción:* Permite a un usuario formal o proveedor eliminar sus propios datos en la base de datos de la página de modo que su información quede eliminada para poder hacer uso de las distintas funcionalidades de ServiApp que requieran del continuo almacenamiento de estos.

- *Acciones iniciadoras y comportamiento esperado*:

Si el usuario desea eliminar sus datos de la página (es decir, eliminar su cuenta de ServiApp), encontrará está opción dentro de las opciones de su perfil y la seleccionará. Esto desencadenará un evento desde la página hasta el código en el cual se le enviará una petición a la base de datos para que ejecute un método que deshabilite la posibilidad de iniciar sesión y la posibilidad de que otro usuario pueda contratar sus servicios (en caso de ser un usuario proveedor). Cabe agregar que estos datos seguirán almacenados en la plataforma ya que seguirán siendo parte del historial e información de contratos de otros usuarios de la plataforma.

Requerimientos funcionales:

- Disponibilidad del enlace al perfil de la cuenta con sesión iniciada en el navegador desde cualquier otra página de la web.
    - Esto implica que el enlace alojado en la opción "Ver mi perfil" este funcionando correctamente.
- Conexión activa entre los archivos que contienen el código fuente de la página y los usados para la conexión con la base de datos (back-end).
    - Esto se resume en el hecho de que el evento provocado al hacer clic en la opción "eliminar cuenta" desde la página de perfil tenga una conexión vigente con el código alojado en el servidor (el computador donde se aloje el código que conecte a la base de datos).
- Disponibilidad de la base de datos para recibir las peticiones por y ejecutar los respectivos métodos respecto estos datos.
    - Esto se resume en el hecho de que exista la conexión estable y vigente con la base de datos y que esta esté funcionando plenamente. En dado caso de que el evento no pueda ser ejecutado correctamente por parte del usuario, se le informará a través de un aviso en pantalla y así mismo se le enviará un mensaje al equipo desarrollador para corregir este error lo más pronto posible.

Inicio de sesión por parte de usuarios registrados a la página web.

- *Descripción:* Permite a los usuarios (formales o proveedores) cuyos datos ya están almacenados en la base de datos iniciar sesión en la página web para hacer uso de ciertas características que lo requieran.

- *Acciones iniciadoras y comportamiento esperado*:

El usuario visitante de la página web al hacer clic en el botón de "inicia sesión" será redirigido a otra ventana en donde se encontrará una lista de selección de correos electrónicos con sesión iniciada en el navegador o en su defecto se le redirigirá

a un formulario de inicio de sesión a un correo electrónico (este apartado es proveído por el API de Firebase implementada). Al finalizar este apartado se compararán los datos obtenidos respecto al correo logueado con los datos alojados en la base de datos, y si estos coinciden con los datos de un objeto de tipo usuario o proveedor el usuario se encontrará identificado en la plataforma con sus respectivos datos (alojados en la base de datos) lo que le permitirá acceder a ciertas características tales como requerir un servicio (realizar contratos) o simplemente revisar su historial de servicios adquiridos y su información de perfil.
.
Requerimientos funcionales:

- Disponibilidad del formulario para los usuarios no formales:
    - Un usuario no formal es cualquier usuario no identificado al hacer uso de la página web, si el formulario es visible y disponible para él significa que documento HTML correspondiente a esta página funciona correctamente.
- Funcionamiento correcto del formulario y conexión entre el código fuente de la página y la base de datos:
- Disponibilidad del enlace al perfil de la cuenta con sesión iniciada en el navegador desde cualquier otra página de la web de Servi-App en cuestión:
    - El botón iniciar sesión deberá estar disponible y ser funcional desde cualquier ubicación dentro de las paginas contenidas en la web de ServiApp, esto quiere decir que no importa si el usuario se encuentre en la página de inicio o en la página correspondiente al listado de proveedores el botón deberá de funcionar correctamente.

Visualización del listado de proveedores y los servicios disponibles para contratar.

- *Descripción:* Permite a cualquier tipo de usuario (formales, proveedores o informales) acceder al listado de proveedores y servicios disponibles para ser contratados. Este listado dependerá del correcto funcionamiento de la base de datos y así mismo se tendrá en cuenta un sistema de calificación (disponible para los usuarios ya registrados únicamente) que servirá para organizar este listado de modo que se promocionen los mejores y más puntuados proveedores y servicios.

- *Acciones iniciadoras y comportamiento esperado*:

Dentro de la página de inicio de la web existirá un botón disponible para cualquier usuario que entre a la página principal en donde al hacer clic redirigirá al usuario a un listado completo y filtrable (por tipo de servicios o calificación) de los proveedores y servicios disponibles para su contratación. Esta información será proveniente directamente de la base de datos. Al momento de acceder a dicha página se ejecutará un script que además de hacer una solicitud para acceder a estos datos almacenados en la base de datos también se encargará de ordenarlos y mostrarlos en pantalla (haciendo uso de herramientas de tipo front-end, es decir, código Javascript, Html y Css ejecutado directamente en el navegador) esta organización se dará en base principalmente a un algoritmo de ordenamiento que priorice la calificación de cada servicio y proveedor y que así mismo sea capaz de filtrar las distintas opciones dependiendo del tipo de servicio ofrecido (filtros elegidos manualmente por el usuario) y/o horario de disponibilidad del proveedor de dicho servicio.

Requerimientos funcionales:

- Disponibilidad del enlace en la página de inicio.
- Conexión estable y vigente desde el código fuente de la página en cuestión con el código ejecutado en el servidor y la base de datos:
    - Dicha conexión se sustenta en el correcto funcionamiento del script responsable de realizar la solicitud a la base de datos y la respuesta correcta de este. Ello también implica que el script responsable de la visualización de la información funcione correctamente ya que es a partir de esto último que se comprueba el correcto funcionamiento de la página.
- Correcto funcionamiento del algoritmo de ordenamiento y así mismo del sistema de puntuación de proveedores y servicios:
    - Este punto es bastante importante ya que este sistema debe de ser inalterable por parte de los proveedores (ellos mismo no pueden votar para sí mismos ni tampoco pueden manipular la calificación recibida) cosa que tiene que ser monitoreada y auditada durante el desarrollo y sobre todo durante y después del lanzamiento del proyecto.
- Correcta clasificación de cada servicio de modo que los filtros aplicables por el usuario para la visualización de la lista funcionen correctamente.

Selección y contratación de un servicio o proveedor por parte de un usuario formal.

- *Descripción:* Permite a un usuario formal elegir cualquiera de las opciones mostradas en la visualización de servicios y proveedores y solicitar un servicio (contrato) (esto sujeto a la disponibilidad dada por el proveedor).

- *Acciones iniciadoras y comportamiento esperado*:

Una vez el usuario este en la sección del listado de servicios y proveedores, si este se encuentra con la sesión iniciada podrá acceder a cualquiera de estos y elegir la opción “contratar”, esto inicializará un evento en la página que desencadenará un método en donde: 1ro se verifique la disponibilidad del

proveedor (en caso de no estar disponible el proceso de contratación quedará en estado de espera), esto abrirá un chat con el proveedor en donde se especificarán ciertos aspectos de conveniencia para ambos, una vez se llegue a un acuerdo existirá la opción de "confirmar contrato", al seleccionar esta opción se ejecutará otro método en la plataforma el cuál almacena en un objeto de la clase "contrato" variables como el precio (establecido previamente), el horario definido (cita), los actores de dicho contrato (nombres del usuario formal y proveedor como participantes del contrato) y el estado actual del contrato que tendrá 2 (por ahora) estados: En proceso (durante) y finalizado (después), este estado cambiará a su ultimo estado una vez el cliente califique al proveedor de dicho servicio (lo que hace que la calificación sea necesaria para continuar con el proceso de la plataforma y asegura que el sistema de puntos funcione de mejor manera.)

Requerimientos funcionales:

- Como requerimiento principal para el correcto funcionamiento de esta función se requiere que la función "Visualización del listado de proveedores y los servicios disponibles para contratar" funcione correctamente ya que esta función se desprende directamente del funcionamiento de esta otra.
- Disponibilidad de la base de datos:
    - Como bien se ha repetido, este es un requerimiento funcional clave para todas las funcionalidades importantes de este proyecto. Para este caso se requiere de esto debido al uso de datos tanto del usuario formal solicitante del servicio como los del usuario proveedor del respectivo servicio para la ejecución del proceso de contratación en todas sus fases.
- Disponibilidad de una herramienta de chat o de comunicación entre el usuario formal y el proveedor:
    - Se requiere del desarrollo de una herramienta de chat en tiempo real o en su defecto de la implementación de una herramienta que permita satisfacer esta necesidad clave dentro del proceso de acuerdo y contratación del servicio

---

Almacenamiento de los contratos en forma de historial de contratos.

- *Descripción: Le permite a un usuario (sea proveedor o formal) visualizar un listado de los contratos concluidos en orden cronológico, esta opción estará disponible para cualquier usuario con sesión iniciada en la sección de "perfil" y estará organizada mediante una estructura de pila.*

- *Acciones iniciadoras y comportamiento esperado*:

Una vez el usuario formal o proveedor se encuentre logueado en la página podrá acceder al apartado de "Historial", ubicado en la sección de navegación de la página, una vez de click a este botón de "Historial" se desplegará un listado de contratos efectuados por este usuario.

Requerimientos funcionales:

- Disponibilidad de la base de datos:
    - Para ello el código fuente encargado de proporcionar la conexión entre la interfaz y la base de datos no debe tener conflicto u error alguno que impida su correcto comportamiento.
    - La base de datos deberá estar en pleno funcionamiento y capacidad, para ello la API y la base de datos implementada de Firebase deberá estar debidamente actualizada y accesible.

IV. AVANCE EN LA IMPLEMENTACIÓN DE LA INTERFAZ DE USUARIO

La interfaz gráfica de usuario se compone de diferentes vistas, en las cuales se puede evidenciar la funcionalidad de las estructuras entre las destacamos:

- Inicio de sesión: realizado con la autenticación de Google.
- Home Page: direcciona a las publicaciones de cada servicio disponible.
- Posts: visualización de todas las publicaciones de proveedores referentes a un servicio.
- Historial: visualización de contratos activos.

La interfaz gráfica hace uso componentes de React.Js , estos componentes contienen código en HTML, CSS y Bootstrap, para definir la estructura a mostrar de la página, al igual que JavaScript y sintaxis de conexión con Firebase, para la interactividad y conexión con la base de datos.

V. ENTORNOS DE DESARROLLO Y DE OPERACIÓN

El software está desarrollado en base al entorno de desarrollo node.js usando el lenguaje de programación JavaScript.
Gracias a la conectividad con la base de datos en línea, el entorno de operación esta basado en el soporte de React.Js y Firebase.
Así, mismo para la interfaz se utilizaron herramientas de desarrollo web tales como HTML5 , Bootstrap4 y CSS3.
El sistema operativo en el cual fue desarrollado este prototipo es Windows 10 y en cuanto a hardware usado para correr las principales pruebas con las estructuras de datos contiene las siguientes características:

- Fabricante: Acer
- Modelo: Nitro AN515-52
- Procesador: Intel®Core ™ i5-8300H CPU @ 2.30GHz

- Memoria Ram: 16.0 GB DDR4 SDRAM
- GPU: Gtx 1050 4gb
- Sistema operativo: Windows 10 - 64 Bits.

VI. DESCRIPCIÓN DEL PROTOTIPO DE SOFTWARE

El prototipo organizado de la manera solicitada y ubicado en una rama única para su entrega. se encuentra en el repositorio de GitHub:
https://github.com/VbarraganP/ProyectoEstructuras

En este prototipo podemos observar la visualización de servicios disponibles, los usuarios registrados en la página, el historial de contratos y adicionalmente la implementación del Heap en los proveedores correspondientes a cada servicio disponible, la cual nos muestra los proveedores organizados de forma descendente de acuerdo con el puntaje de cada proveedor.

VII. PRUEBAS DEL PROTOTIPO

Para la realización de las pruebas se utilizaron 5 funcionalidades del software implementadas en diferentes estructuras para realizar un análisis comparativo.

- Creación de Usuarios (Listas) y Proveedores (Heap).
- Actualización de un proveedor en Heap (ChangePriority()) y Listas.
- Extracción del Proveedor con mayor calificación en Heap (ExtractMax()).
- Consulta total del Historial (Listas) y Proveedores por servicio (Heap).
- HeapSort() (Heap).

Se realiza un análisis de los resultados en la siguiente sección.

I. ANÁLISIS DE COMPARATIVO

Se presentan las tablas y gráficos correspondientes a las distintas funcionalidades. Ver anexos [3].

-Funcionalidad 1, Creación de Usuarios y Proveedores:
Se utilizó una librería de JavaScript denominada "Faker", encargada de generar datos de prueba en todas las funcionalidades. Al medir la creación de usuarios encontramos que el supuesto de la linealidad no es claro por el ajuste realizado (Valor R-Squared) y lo asociamos al funcionamiento de la librería donde la complejidad de esta funcionalidad parece ser cuadrática $O(N^2)$. Para la creación de Proveedores se evidencia una complejidad lineal O(N). Sin embargo, los datos tipo Proveedor que se generan son más pesados (más atributos) y esto genera que los tiempos sean mayores que los registrados en usuarios.

-Funcionalidad 2, Actualización de un proveedor:
Luego de realizar la inserción de datos tipo Proveedor en la lista doblemente enlazada y el Heap se accede a las estructuras para actualizar el dato N-1. En la lista Doblemente Enlazada se recorre toda la lista hasta llegar al penúltimo dato y modificarlo, la anterior acción tiene complejidad O(N). Para el Heap y su función ChangePriority(), se accede al dato directamente gracias a su posición que en este caso sería el tamaño-1, se modifica y posteriormente se reordena la estructura, esta acción tiene complejidad constante O(1).

-Funcionalidad 3, Extracción del Proveedor con mayor calificación en Heap (ExtractMax()).
Al realizar la inserción de Proveedores en el Heap estos son organizados con respecto a su calificación, los mejores calificados estarán situados en las primeras posiciones del arreglo donde está implementado el árbol. La función ExtractMax() accede a la primera posición del arreglo, extrae, retorna al proveedor y reorganiza la estructura en un tiempo constante O(1).

-Funcionalidad 4, Consulta total del historial y Proveedores por servicio:
Se realizó la creación de los diversos contratos que fueron insertados en la pila del Historial, se comprobó la linealidad de la operación de "recorrido" y acción (mostrar en consola) para una pila. La complejidad de la Consulta del historial es de O(N). En el caso de los proveedores, se debe realizar una operación de complejidad constante (ExtractMax()) por el número de datos N, se presentó un problema con más de 500000 datos y se atribuye a la cantidad de atributos de proveedor. Esta operación de consulta tiene una complejidad lineal O(N).

-Funcionalidad 5, HeapSort():
Esta funcionalidad está enfocada a la consulta y ordenamiento simultáneos que necesita Servi-App en su interfaz gráfica, retornando un arreglo ordenado por la calificación de sus proveedores. Su complejidad es lineal O(N).

En resumen, presentamos una tabla con las distintas funcionalidades, la estructura donde están implementadas, la cantidad de datos, tiempo ejecución en milisegundos y su notación Big(O).

Nombre de la funcionalidad	Tipo(s) de estructura de datos	Cantidad de datos probados	Análisis realizado (Notación Big O)	Tiempos de ejecución (ms)
HeapSort	Heap	10.000	O(N)	5.55
		100.000	O(N)	10.38
		500.000	O(N)	41.73
		1.000.000	O(N)	94.34
Extracción del Proveedor con mayor calificación	Heap	10.000	O(1)	15.49
		100.000	O(1)	15.51
		500.000	O(1)	15.06
		1.000.000	O(1)	15.22
Actualización Cambiar prioridad	Heap	10.000	O(1)	0.16
		100.000	O(1)	0.16
		500.000	O(1)	0.16

		1.000.000	O(1)	0.16
Actualización	Double Linked List	10.000	O(N)	0.79
		100.000	O(N)	3.36
		500.000	O(N)	9.88
		1.000.000	O(N)	20.83
Consulta Total Historial	Stack	10.000	O(N)	8148
		100.000	O(N)	78357
		500.000	O(N)	452288
		1.000.000	O(N)	992146
Consulta Total Proveedores	Heap	10.000	O(N)	21899
		100.000	O(N)	215719
		500.000	O(N)	1282077
		1.000.000	O(N)	--
Creación	Double Linked List	10000	$O(N^2)$	76.41
		100000	$O(N^2)$	496.11
		500000	$O(N^2)$	2433.71
		1000000	$O(N^2)$	3982.18
Creación	Heap	10000	O(N)	4148
		100000	O(N)	39757
		500000	O(N)	218088
		1000000	O(N)	400611

VIII. DIFICULTADES Y LECCIONES APRENDIDAS

Principalmente, las dificultades y lecciones aprendidas con el desarrollo del proyecto fueron:

- La creación de un proyecto en lenguajes y tecnologías sobres las cuales solo se tienen conocimientos básicos, añade una dificultad extra la cual requiere de una mayor inversión de tiempo.
- Intentar implementar todo tipo de estructuras en un proyecto, puede ser contraproducente, siempre es necesario hacer un análisis para determinar la opción más conveniente a seguir.
- La conexión con bases de datos que actúan en tiempo real requiere de un adecuado manejo de procesos síncronos y asíncronos.
- El manejo adecuado de las tecnologías, junto documentación actualizada brinda mayor eficiencia en el desarrollo del proyecto.

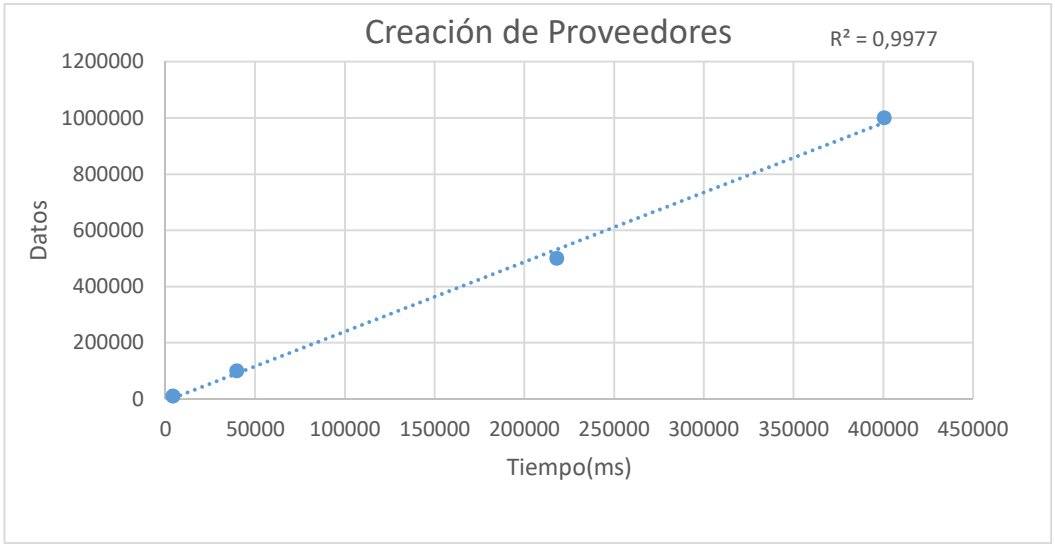
ANEXOS

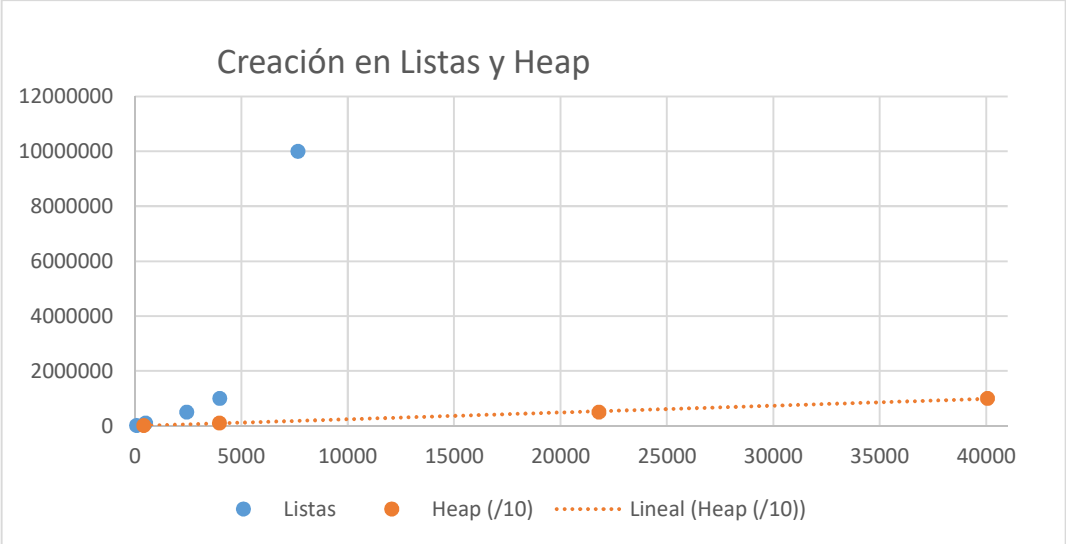
[3] Tabla de Datos y Gráfico de Complejidad.
1) Funcionalidad: Creación de Usuarios (Listas).

Tiempo (ms)	Datos
76.412501	10000
496.115601	100000
2433.7192	500000
3982.1804	1000000
7667.0175	10000000

Creación de Proveedores (Heap)

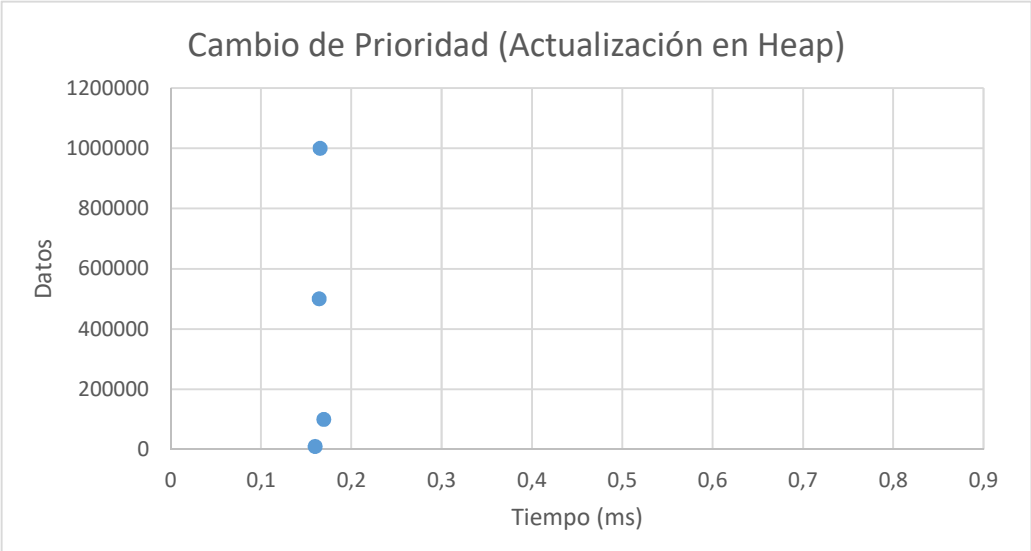
Tiempo (ms)	Datos
4148.5371	10000
39757.2595	100000
218088.3466	500000
400611.4691	1000000





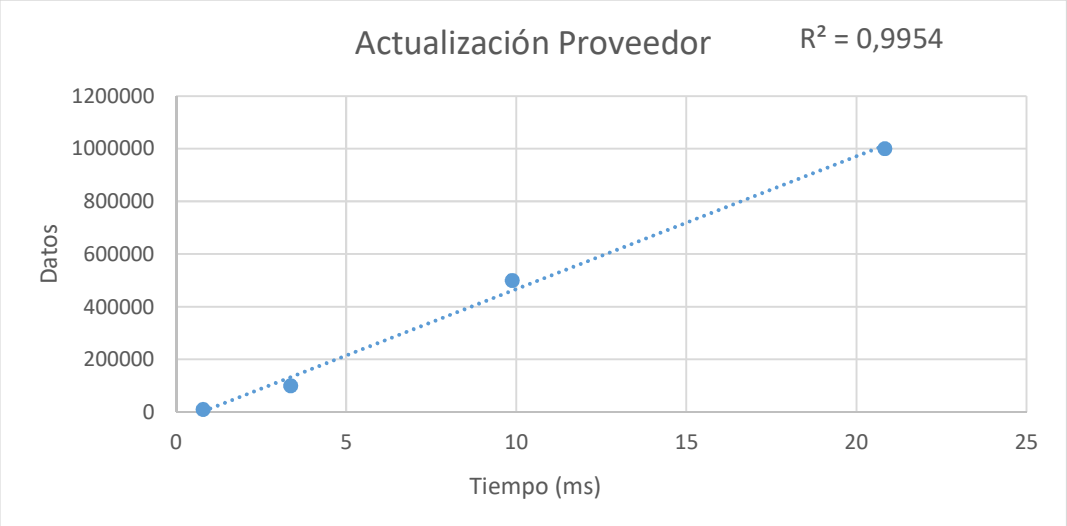
2) Funcionalidad: Actualización de un Proveedor (ChangePriority()) en Heap y Listas.

Tiempo (ms)	Datos
0.16003	10000
0.169499	100000
0.164479	500000
0.165323	1000000

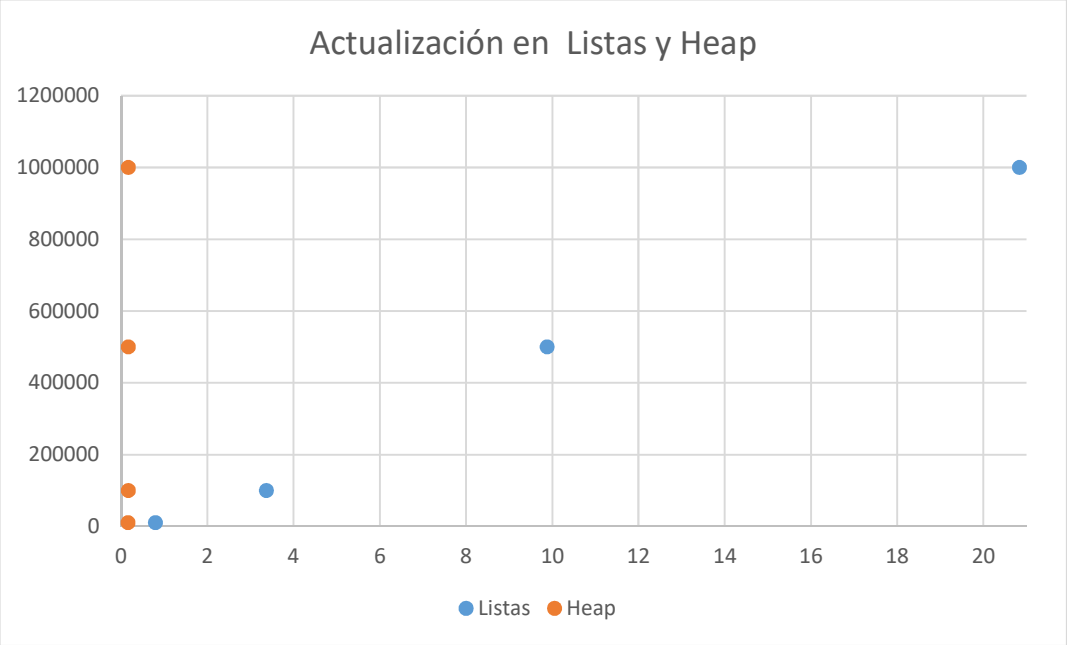


Actualización de un Proveedor (Listas).

Tiempo	Datos
0.794099	10000
3.3673	100000
9.8837	500000
20.837	1000000

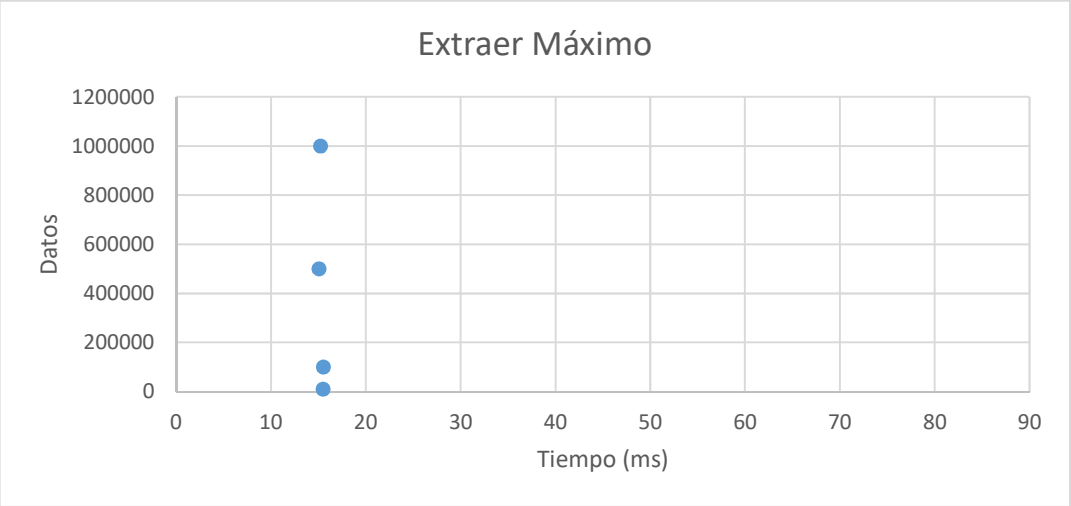


Comparación de Estructuras:



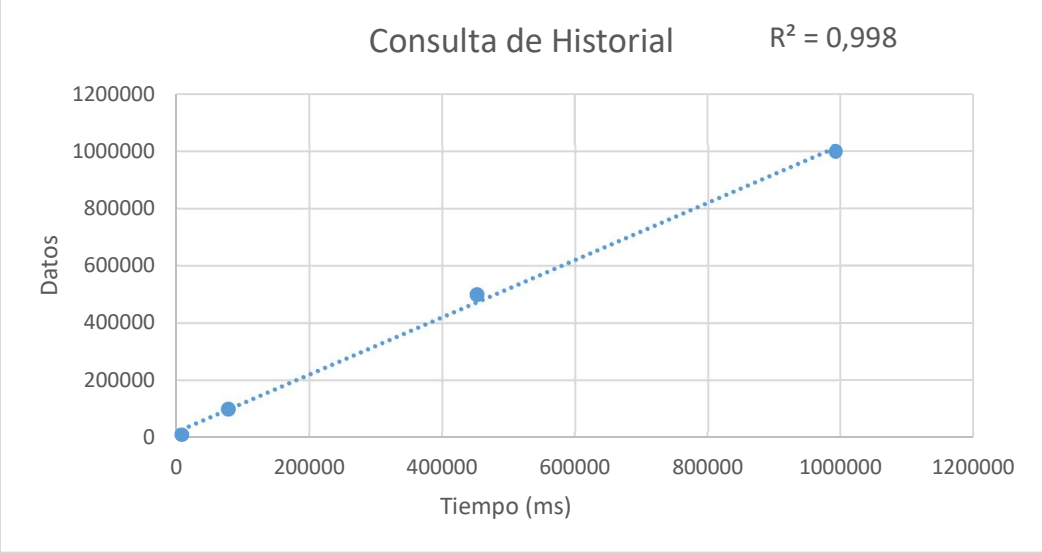
3) Funcionalidad: Eliminación del último Proveedor utilizando ExtractMax().

Tiempo(ms)	Datos
15.4981	10000
15.519701	100000
15.063999	500000
15.226399	1000000



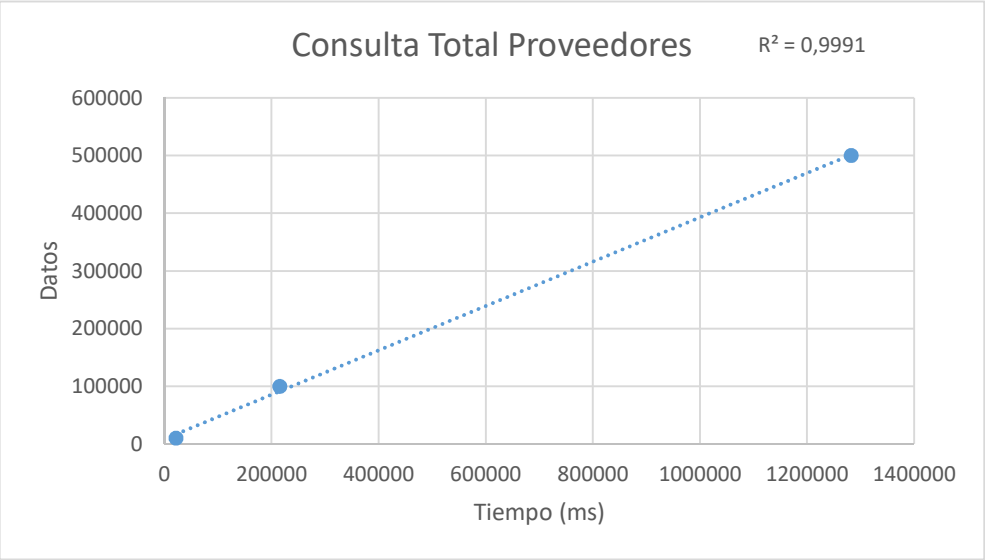
4) Funcionalidad: Consulta total del historial (Listas).

Tiempo (ms)	Datos
8148.6185	10000
78357.8260	100000
452288.05	500000
992146.597	1000000

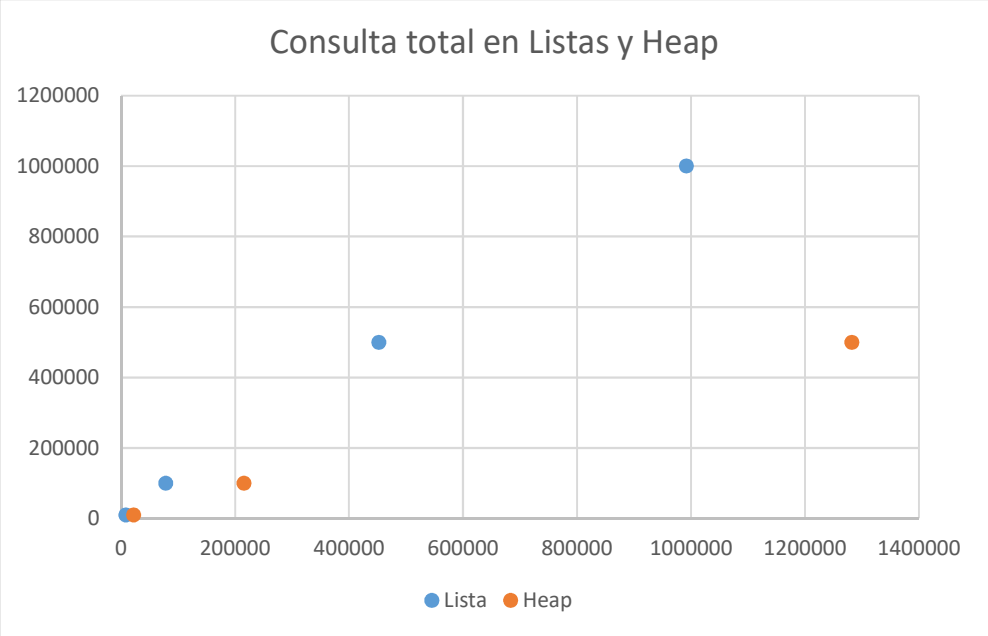


Consulta Total de Proveedores por servicio (Heap).

Tiempo (ms)	Datos
21899.7611	10000
215719.9823	100000
1282077.58	500000



Comparación de Estructuras:



5) Funcionalidad: HeapSort()

Tiempo (ms)	Datos
5.553399	10000
10.3892	100000
41.7389	500000
94.340099	1000000