Poptr

Santiago Suárez Aguilar, Nicolas Arevalo Rodriguez, Angel Gonzalez Bejarano

No. de Equipo Trabajo: 3

I. INTRODUCCIÓN

En el presente documento se describe una alternativa de solución a un problema común en la cotidianidad, el cual está asociado al proceso que conllevan los trámites en Colombia. Debido a la burocracia y la baja eficiencia en términos de tiempo para el desarrollo de los mismos, se convierten en una tarea tediosa y demandante. Durante el documento, se describirán los principales requerimientos del software, la alternativa de solución propuesta con sus especificaciones técnicas incluyendo un análisis y prueba de la misma.

II. DESCRIPCIÓN DEL PROBLEMA A RESOLVER

Según el BID por sus siglas, Banco Interamericano de Desarrollo, en el año 2018 en Colombia se necesitaban 7.4 horas para completar un trámite, situando a nuestro país en el top tres de países latinoamericanos con más tiempo en realizar un trámite [1].

El porcentaje de trámites completamente digitales en nuestro país es del 4%, una cifra que está en el promedio de la región situándose esta en 3.7% [1]. Aun así, existe todavía mucho potencial para implementar la tecnología en la simplificación de trámites, asumiendo que ya el Estado mismo ha hecho una simplificación de manera legal, es decir, ha reducido la complejidad de los trámites y los requerimientos que se suponen para ellos.

Estamos hablando de que el potencial que tienen los trámites digitales en nuestro país es del 35%, es decir, trámites que se pueden empezar y completar en línea.

Mientras esto ocurre, y la brecha digital que en nuestro país existe, se reduce, lo que se plantea con esta idea es optimizar los trámites presenciales ya existentes y darles las herramientas para poder simplificarlos, conectando a los usuarios que realizan estos trámites de una manera fácil, sencilla y eficaz.

III. USUARIOS DEL PRODUCTO DE SOFTWARE

El software contará con dos tipos de usuarios, los cuales se caracterizan dependiendo de los privilegios de acceso a la información y a las diferentes funcionalidades del software. Por un lado estarán las personas que quieren agendar un trámite o interactuar con otros usuarios a través de la red social, ellos tendrán acceso a la información publicada por los demás, además podrán agendar turnos, crear publicaciones y comentar las mismas. Por otro lado, estarán los administradores, los cuales podrán acceder a las colas del agendamiento de turnos, podrán eliminar contenido que consideren erróneo o inoportuno, además de manipular las listas de usuarios. En las colas podrán eliminar turnos agendados.

IV. REQUERIMIENTOS FUNCIONALES DEL SOFTWARE

- *Registro de usuarios:* Recopilación de datos personales y de contacto que faciliten la interacción con el usuario. El usuario debe proporcionar datos verídicos con el fin de facilitar el contacto con el mismo. En el caso de que el usuario incorpore datos con un formato erróneo, se indicará que la entrada no es válida.

- *Reporte de usuarios:* Listado de todos los usuarios registrados, esta funcionalidad le permite al administrador conocer los datos de sus usuarios

- *Búsqueda de usuario*: Búsqueda parcial de un usuario a partir de alguno de sus atributos, retornará el nodo con la información del usuario en cuestión

- *Actualización de usuario:* Búsqueda del usuario a partir de uno de sus atributos, y además se asigna un nuevo valor al atributo que se desee cambiar.

- *Eliminación de usuario*: A partir del índice en el cual se encuentra alojado el nodo del usuario, se puede eliminar dicho nodo de la lista. Es necesario conocer la posición del usuario previamente.

- *Validación de usuarios:* Consulta y verificación en la base de datos de que el usuario exista. Es necesario conozca sus credenciales para acceder al sistema.

- *Creación de publicación:* Al ser una red social, es necesario establecer una funcionalidad que le permita a los usuarios publicar noticias, preguntas o información pertinente que corresponda a la temática deseada.

- *Eliminación de publicación:* Esta funcionalidad responde a la necesidad de quitar publicaciones que contengan información errónea, desactualizada o contenido inoportuno por parte del usuario. Esta funcionalidad estará disponible para los administradores, que bajo sus criterios y normas de uso del sitio, establecerá la validez de la publicación.

- *Búsqueda de palabra en publicación:* Búsqueda parcial de palabras en la publicación para obtener información que el usuario quiere obtener sobre un trámite o alguna funcionalidad del sitio.

- *Agendamiento de turno:* Los usuarios podrán agendar turnos para los trámites disponibles dentro del sitio, en esta sección se podrá seleccionar el horario en el cual desean asistir a realizar su proceso.

- *Eliminación de turnos en la cola:* Para depurar el agendamiento de turnos, una vez pase el horario asignado al turno, el administrador podrá eliminar el turno en cola, ya sea por inasistencia o porque ya se realizo el tramite. Esta opción también se habilitará de manera opcional en el caso de que no se pueda ofrecer el servicio.

V. DESCRIPCIÓN DE LA INTERFAZ DE USUARIO PRELIMINAR

Se hará uso de una interfaz gráfica que facilitará la interacción del usuario con el software y los demás usuarios. La pantalla inicial permitirá que el usuario se registre o en el caso de que esté registrado, valide su sesión. En la pantalla de registro, el usuario podrá introducir sus datos, los cuales serán usados para la validación de la sesión y además para contactarlo en caso de que agende un turno.

Una vez el usuario valide su sesión podrá acceder a la sección de publicaciones y trámites. En la sección de trámites el usuario podrá crear publicaciones, y también podrá comentar las publicaciones de los demás.

En la sección de trámites el usuario podrá ver un listado de trámites que puede realizar, para de esta forma elegir el trámite para el cual desea agendar el turno. Por otro lado, el usuario también podrá observar los trámites que están en curso (agendados).

Finalmente, en el caso de la persona que tiene el rol de administrador, esta tendrá habilitada una sección adicional, la cual responde a el manejo de la cola de turnos agendados, en esta se podrá eliminar turnos de la cola. Adicionalmente, tendrá habilitada la opción de eliminar publicaciones.

El mockup de la interfaz gráfica descrita se puede ver en el siguiente link:
https://xd.adobe.com/view/5e4308ae-9b6b-4553-647c-fae6f52ea6a8-2872/

VI. ENTORNOS DE DESARROLLO Y DE OPERACIÓN

El software se desarrolló en Typescript y la interfaz gráfica se realizó a partir de una librería de Javascript llamada React. El IDE elegido fue Visual Studio Code y además como entorno de ejecución se usó Node.js. El servidor creado como *local host* se aloja en el navegador Google Chrome durante el tiempo de ejecución.

El sistema operativo en el que se desarrolla la operación es Windows 10 y las especificaciones del hardware son:

- Procesador: Intel Core i7-9750H 2.6GHz
- RAM: 16 GB DDR4

VII. PROTOTIPO DE SOFTWARE INICIAL

El prototipo de software inicial está compuesto por dos partes. Inicialmente se tiene la implementación de una primera versión de la interfaz de usuario, a la cual se puede acceder leyendo las instrucciones de acceso en el repositorio; en este prototipo la interfaz no tiene relación directa con la implementación de las estructuras de datos. Por otro lado, se realizó la implementación de una lista simplemente encadenada con cola de tipo genérico; cada lista está compuesta entonces por nodos que poseen una data de tipo cualquiera T y la dirección del siguiente nodo en la lista. Para este caso, la implementación de la lista está dirigida a objetos de tipo User, estos objetos tienen los atributos de userName (nombre de usuario), userDescription(descripción de usuario), userPhoto(foto de usuario) y user(email), los cuales se importan para la realización de las pruebas desde archivos .json que contienen los arreglos de objetos para los diferentes tamaños. Estos archivos se encuentran en la carpeta *data*, de la carpeta *src* del repositorio. Teniendo en cuenta lo anterior, cada nodo de la lista contendrá un “usuario” con sus respectivos atributos.

A continuación, se describen brevemente las operaciones funcionales que soporta la lista:

- Creación de la lista: los valores defecto de la cabeza y cola de la lista son **null**
- **pushFront = (data: T)**: Permite insertar un nuevo nodo a partir de un dato en la cabeza de la lista
- **pushBack = (data: T)**: Permite insertar un nuevo nodo a partir de un dato en la cola de la lista.
- ***items()** : Permite recorrer la lista.
- **traverse()** : Permite consultar todos los datos de la lista haciendo un recorrido desde la cabeza.
- **search = (sdata :T)** : Permite buscar un dato en la lista. En este caso, se puede buscar un usuario a partir de su nombre, foto, descripción o email. Esta operación retorna el nodo al que corresponde el dato e imprime en consola los datos correspondientes a ese nodo.
- **updateNode = (prevdata : T, newdata: T)** : Permite la actualización de un dato. Para este caso, la función busca el dato y verifica si corresponde a userName, userPhoto, userDescription o user, y luego cambia su valor respectivo por el nuevo dato que se tiene. Para este caso, la operación imprime en consola el conjunto de datos del usuario con el nuevo dato.
- **removeNode(index : number)**: Permite remover un nodo de la lista a partir de la position en que se encuentra estrictamente en la lista. Además de

remover el nodo de la lista, esta operación muestra los datos del nodo que se ha removido de la lista.

- **lenght()** : Permite obtener el número de elementos en la lista. En este caso, retornaría el número de usuarios registrados.

Para la implementación de la estructura de datos, se importan los respectivos conjuntos de datos con los que se quiera trabajar en el archivo Users.js de la carpeta *classes,* ubicada en *src* y se exportan los procedimientos que se quieren mostrar en consola, así, en el archivo landing.tsx perteneciente a la carpeta *Components,* se puede ver como se importa el método perteneciente a Users.js y se utiliza posteriormente para mostrar los resultados en consola. Teniendo en cuenta esto, para la prueba de 10 mil datos, por ejemplo, se exporta una función que crea una instancia de la linkedlist como la lista de usuarios, toma los objetos del archivo userData.json y los inserta a la lista encadenada y luego realiza las diferentes operaciones sobre la lista a conveniencia.

Se debe tener en cuenta que para esta ocasión todas las operaciones sobre los datos se realizan en tiempo de ejecución y los resultados se muestran por consola.

A continuación se encuentra el enlace al repositorio:

- **https://github.com/angegonzalez/Poptr**

VIII. PRUEBAS DEL PROTOTIPO

Las funcionalidades a probar se implementaron en la lista creada para los usuario. Para dichas funcionalidades se realizaron pruebas con:

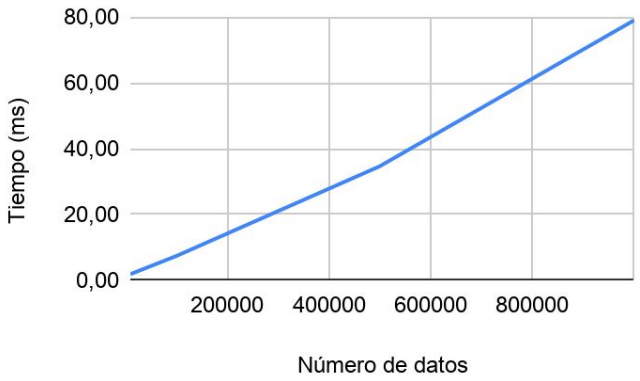
- 10000 datos
- 100000 datos
- 500000 datos
- 1000000 datos

Se omitió la prueba de 10000000 de datos, por las razones que serán descritas en la sección de dificultades. Con el fin de cuantificar los tiempos que toma correr cada funcionalidad, se usaron los métodos console.time() y console.timeEnd(), estos permiten calcular el tiempo que toma correr el código en la consola del navegador.

A continuación se presentan las tablas comparativas y los gráficos para cada una de las funcionalidades elegida para tal fin:

❖ Funcionalidad: Registro de usuarios (Fill)

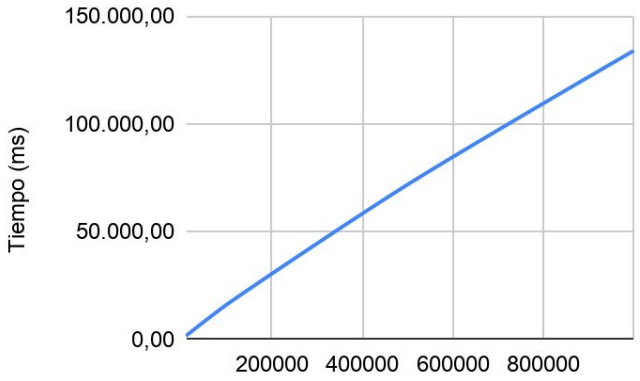
Número de datos	10000	100000	500000	1000000
Tiempo (ms)	1,89	7,39	34,67	79,03



Se observa que el tiempo que toma llenar la lista de usuarios es lineal ya que se debe repetir la operación pushBack() n veces. Por lo anterior se puede concluir que el tiempo que toma la operación pushBack() es constante, es decir, pushBack() es de complejidad O(1).

❖ Funcionalidad: Reporte de usuarios (Traverse)

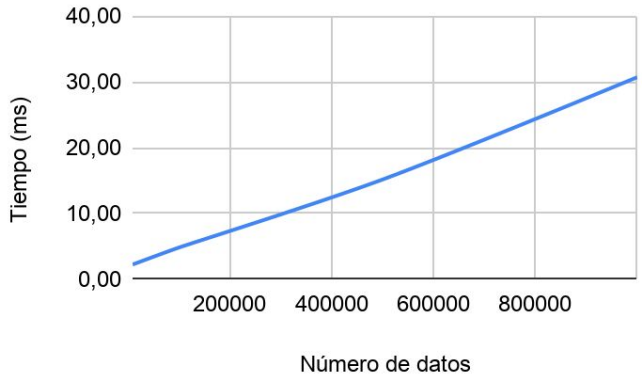
Número de datos	10000	100000	500000	1000000
Traverse	1.661,87	16187,48	71.907,57	134055,75



Se observa que en esta funcionalidad al ser necesario recorrer toda la lista para acceder e imprimir la información de los nodos, el tiempo que toma es lineal.

❖ Funcionalidad: Búsqueda de usuario (Search)

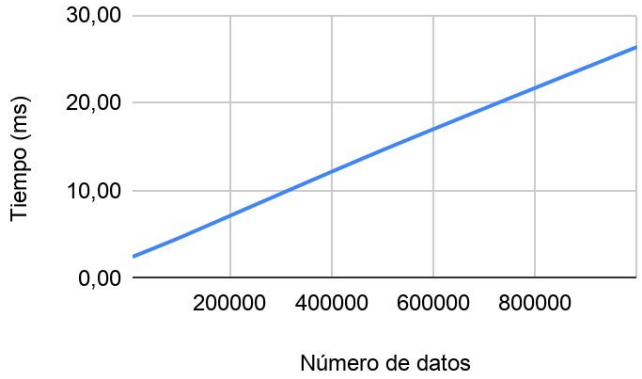
Número de datos	10000	100000	500000	1000000
Tiempo (ms)	2,25	4,81	15,15	30,71



Se observa que en esta funcionalidad al ser necesario recorrer la lista para acceder al nodo que se busca, el tiempo que toma es lineal. Esta prueba se realizó para el peor de los casos, por lo tanto, el acceso a elementos que estén más cercanos a la cola o a la cabeza tomarán un tiempo menor.

- Funcionalidad: Actualización de usuario (Update)

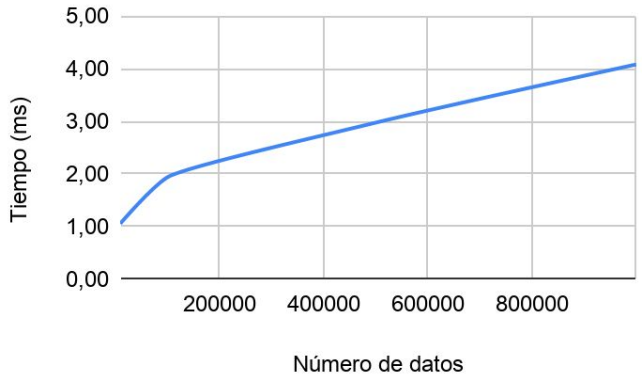
Número de datos	10000	100000	500000	1000000
Tiempo (ms)	2,46	4,58	14,60	26,37



Se observa que en esta funcionalidad al ser necesario recorrer la lista para acceder al nodo que se busca, el tiempo que toma es lineal. En comparación a la funcionalidad anterior, se puede resaltar que toma un poco más de tiempo debido a que además de realizar la operación de búsqueda, tiene que realizar la operación de actualizar.

- Funcionalidad: Eliminación de usuario usando índice (Delete)

Número de datos	10000	100000	500000	1000000
Tiempo (ms)	1,06	1,94	2,98	4,08

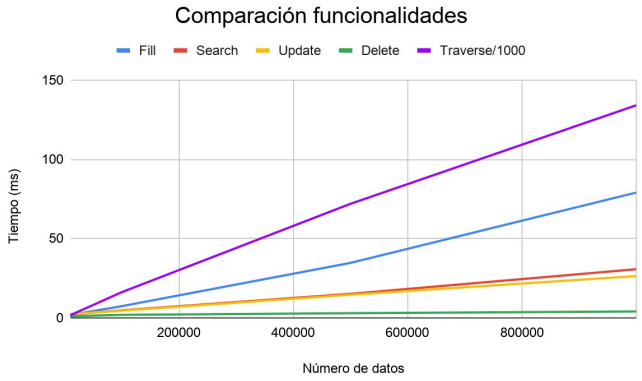


Se observa que en esta funcionalidad al ser necesario recorrer la lista para acceder al nodo que se busca a partir del índice, el tiempo que toma es lineal. La prueba se realizó para el peor de los casos.

I. Análisis de Comparativo

A continuación se observa una tabla comparativa de los tiempos de cada funcionalidad para diferentes cantidades de datos, el tiempo está expresado en milisegundos:

Número de datos	10000	100000	500000	1000000
Fill	1,89	7,39	34,67	79,03
Search	2,25	4,81	15,15	30,71
Update	2,46	4,58	14,60	26,37
Delete	1,06	1,94	2,98	4,08
Traverse	1.661,87	16187,48	71.907,57	134055,75



Nota: Los tiempos de la funcionalidad traverse se dividieron en 1000 para lograr observar la comparación de forma más clara.

De la tabla y la gráfica se puede concluir, que la funcionalidad con mayor costo operacional es generar el reporte de todos los

usuarios (traverse), ya que se debe acceder a cada nodo y además imprimir la información contenida en el mismo. Además también se observa, que la mayoría de operaciones sobre una lista enlazada son de orden lineal, es decir, complejidad O(n). El resumen de la complejidad operacional en la notación Big O de cada una de las funcionalidades se encuentra a continuación:

Funcionalidad	Complejidad
Fill	O(n)
Search	O(n)
Update	O(n)
Delete	O(n)
Traverse	O(n)

II. ROLES Y ACTIVIDADES

INTEGRANTE	ROL(ES)	ACTIVIDADES
Santiago Suárez Aguilar	Coordinador	● Se encargó de programar las reuniones para el desarrollo del proyecto ● Promovió una comunicación activa y asertiva
	Investigador	● Se encargó de orientar preguntas con respecto al desarrollo del proyecto, tanto del prototipo esperado como el documento
Angel Gonzalez Bejarano	Experto	● Designó las actividades a desarrollar por los integrantes
	Tecnico	● Aportó con el entorno de desarrollo y el uso del mismo ● Se encargó del desarrollo de la interfaz gráfica
Nicolas Arevalo Rodriguez	Secretario	● Se encargó de verificar que la comunicación se realizará en los canales pertinentes, además de asegurar que se cumpliera lo acordado.
	Observador	● Se encargó de realizar el rol de apoyo durante el desarrollo de las actividades, además verificó el correcto desarrollo de las mismas con el fin de cumplir con los objetivos

III. DIFICULTADES Y LECCIONES APRENDIDAS

Una de las principales dificultades durante el desarrollo fue las pruebas de carga, debido a que el software almacena su información en Firebase (base de datos) para la persistencia de los datos y dicha plataforma no permite almacenar dichos datos sin un costo adicional. Por ello, se deben realizar las pruebas en consola. Por otro lado, también fue necesario omitir la prueba de 10 millones de datos debido a que el heap de Node.js no permite hacer dicha prueba, ya que la cantidad de recursos necesarios para realizar la prueba de carga son muy altos.

Durante el desarrollo del primer prototipo se evidenció la necesidad de hacer un uso eficiente de los recursos y las estructuras de datos para de esta forma optimizar los tiempos que toman ejecutar las funcionalidades del software. Además, se enfatizó en la importancia del trabajo en equipo para lograr cumplir los objetivos planteados con una comunicación asertiva y con tareas precisas y claras basadas en un proceso previo de concepción de la idea.

IX. REFERENCIAS

[1]“BID- El fin del trámite eterno", BID, 2020. [Online]. Available: https://cloud.mail.iadb.org/fin_tramite_eterno?UTMM=Direct&UTMS=Website#el-problema-con-los-tramites. [Accessed: 13- Mar- 2020].