Crowdfunding, Inversiones, Donaciones y Comercio Electrónico en Colombia

Autor: Bedoya Torres Juan Eduardo, González Castro Camilo Andrés, Moreno Hernández José Luis, Silva Capera Daniel Santiago.
Equipo de trabajo No. 3: {Janu}

I. Introducción

En Colombia existen muchas personas capaces y brillantes, motivadas a emprender nuevas ideas, proyectos o se dedican a oficios de manera amateur. De una forma u otra muchas de estas se encuentran con dificultades para cumplir sus objetivos por falta de infraestructura, financiamiento o colaboración para su desarrollo.

Es aquí donde la Ingeniería de Sistemas y Computación con el uso de las tecnologías de la información puede solucionar este problema conectando a estas personas con otras interesadas en ayudarlas a alcanzar sus objetivos a través de la recaudación de fondos y otras herramientas.

Esta iniciativa pretende apoyar al desarrollo de ideas incluso cuando culmine su etapa inicial y a los usuarios que hagan de esto su medio de trabajo o hobby.

II. Descripción del problema a resolver

Debido al decaimiento de la economía nacional agravada por el pánico y aislamiento a causa de la pandemia "Covid-19" [1], muchos negocios existentes y nuevas ideas han tenido que suspenderse o erradicarse. Ante esto el gobierno nacional ha tomado medidas para estimular y reactivar la economía con préstamos y créditos a comerciantes para su sostenimiento. Sin embargo, esto no es precisamente una buena alternativa para muchos de estos, ya que de una u otra forma se trata de un endeudamiento, incluso puede ser "empezar por el pie izquierdo".

Para ello existen diferentes plataformas exitosas recaudación de fondos ("Crowdfunding"), entre ellas *Kickstarter*, un sitio fundado en 2009 de origen estadounidense[2] que motivó a concebir una nueva solución a este problema, ya que esta no termina adaptándose completamente a las necesidades en el país. La filosofía de Kickstarter es recaudar fondos para cualquier tipo de causa siempre que impulse un proyecto o idea novedosa, a cambio a los colaboradores se les hace parte del objetivo alcanzado [3].

Los motivos por los cuales esta iniciativa no ha tenido una gran acogida en Colombia pueden deberse a la desconfianza de plataformas electrónicas, los métodos de pago utilizados y las escasas propuestas en la plataforma originarias dentro del territorio (620 en total, de los cuales 60 han logrado su objetivo) [4].

Más allá de esto, se encontró un gran potencial en el concepto que se ve restringido por su normativa, en especial refrentes a los incentivos a los colaboradores, la prohibición de donaciones a otras causas benéficas o humanitarias [3], la poca visibilidad de servicios ofertados por miembros (ya que se trata de lograr objetivos y no ofertar bienes o servicios).

III. Usuarios del producto de Software

Cualquier persona que haga uso de la aplicación podrá ejercer cualquiera de los dos perfiles al tiempo, ya que se les identifica como personas, y pueden crear proyectos ideas y colectas a la vez que ayudan con el financiamiento de otras. El público que se desea capturar es todo aquel que sea mayor de edad y que pueda acceder de manera sencilla a dispositivos electrónicos y conexión a internet.

IV. Requerimientos funcionales software

Para el primer prototipo del proyecto se estima que su uso sea desde una consola de comandos para probar su funcionalidad e interacción entre usuarios y proyectos. Para ello se definieron dos clases (la clase usuario y la clase proyecto.

A continuación, se describen las funcionalidades del software en su propia sección:

A. Mensaje de bienvenida [Decisión]:

- Descripción: se muestra por consola un mensaje de bienvenida al usuario en el que se pregunta si este es nuevo en la plataforma o no, para proceder a la *creación de usuarios (signup)* o al *inicio de sesión (login)*.
- Acciones iniciadoras y comportamiento esperado:
  1. Esta acción se ejecuta justo al inicio del aplicativo, por lo que no requiere de una acción iniciadora más que empezar la ejecución del aplicativo.
  2. Se despliega un mensaje de bienvenida como el siguiente: "Welcome! Is this your first time here?"
  3. Se espera la entrada del usuario para saber si debe o no registrarse. Dicha entrada se espera que sea "Y" para empezar el registro. De lo contrario se asume que tiene una cuenta para el ingreso.
  4. Se despliega la funcionalidad *login* o *signup*, según sea requerido por el criterio anterior.
- Requerimientos funcionales:
  - Se despliega un mensaje en consola descrito anteriormente.
  - Se espera la entrada "Y" o alguna otra para toma de decisiones.
  - En caso de que la entrada sea numérica o de algún otro tipo se asume que el usuario no es nuevo. Se espera controlar este comportamiento en versiones próximas.

B. Creación de Usuarios (signup) [Creación}:

- Descripción: Se le solicita al usuario si no dispone de alguna forma de acceder, que cree una cuenta valida, para ello se le solicita información crucial como su nombre, contraseña para acceder a su cuenta, etc.
- Acciones iniciadoras y comportamiento esperado:
  1. La acción iniciadora de esta funcionalidad es la funcionalidad inmediatamente anterior, en la que se pregunta si el usuario va a ser nuevo o no.
  2. Se pregunta por un nombre de usuario o *nickname*, contraseña para la cuenta que se creará, su fecha de nacimiento y un email.
  3. Se despliega un mensaje que muestra que el usuario ha sido creado con éxito.
- Requerimientos funcionales:
  - Se consulta a través de líneas de consola por cada campo, como, por ejemplo: "Please enter an username:"
  - Seguido del mensaje para solicitar la entrada, se espera la entrada. Cada una de las entradas son entradas de texto. En el caso de las fechas de nacimiento, no se rectifica por el momento que sea una fecha valida.
  - Se insertan los datos en un archivo JSON que servirá para que la información se preserve.
  - Se despliega el mensaje "you have successfully created a new account!".
  - Se manda al usuario creado directo a la página principal.

C. Inicio de Sesión (Login)[Lectura]

- Descripción: Se pide al usuario ingresar la información que suministró para registrarse y permitir el paso a la página principal.
- Acciones iniciadoras y comportamiento esperado:
  1. La acción inicializadora es la opción de ingresar con un usuario existente de la funcionalidad A.
  2. Se pregunta al usuario por el nombre de usuario y contraseña que tiene a través del mensaje por consola.
  3. Se hace la verificación de las cadenas ingresadas con los nombres de usuario y contraseñas almacenadas en el archivo JSON
  4. Una vez se verifica que los datos ingresados son correctos se pasa a la página principal del usuario.
- Requerimientos funcionales:
  - Se lee el archivo JSON de usuarios antes de empezar cualquier acción.
  - Se consulta al usuario vía consola los campos de nombre de usuario y contraseña.
  - En caso de ingresar datos no validos se vuelve a preguntar por el usuario y la contraseña.
  - Los datos en el archivo JSON se guardan en memoria temporal para hacer las comprobaciones de objetos guardados. Estos se guardarán en memoria permanente justo antes de salir de la sesión
  - No existen parámetros no validos pese a que todo lo que se inserte en consola será tomado como cadena de texto.

D. Edición de usuarios [Edición]

- Descripción: Se le pregunta si al usuario se desea modificar algún parámetro como nombre de usuario o contraseña para que lo haga.
- Acciones iniciadoras y comportamiento esperado:
  1. Una vez creado el usuario, cada vez que retorne a la pagina principal, y este decida no salir y guardar. Se le pregunta si desea editar o no el usuario.
  2. Se le pregunta al usuario que campos desea cambiar (usuario y contraseña).
  3. Una vez termine la edición se vuelve a preguntar para saber si el usuario está satisfecho con los cambios hechos.
- Requerimientos funcionales:
  - Se consulta por medio de la consola si el usuario desea o no cambiar algún campo.
  - La respuesta únicamente debe ser "Y", de lo contrario, se asume que el usuario no desea hacer cambios.
  - Los campos que se desee cambiar pueden ser únicamente el nombre de usuario y la contraseña. Como son cadenas de texto, aunque se ingresen números se tomarán como válidos.
  - Se manda al usuario creado directo a la página principal.
  - Todos los cambios efectuados se realizan en la memoria temporal. Únicamente se guardan los cambios en el momento que el usuario desee guardar y salir.

E. Eliminación de usuarios [Eliminación]

- Descripción: Se eliminan los usuarios de la memoria que se almacena de manera temporal, para posteriormente reflejar el cambio en la memoria permanente.
- Acciones iniciadoras y comportamiento esperado:
  1. Una vez el usuario haya ingresado en la funcionalidad principal, se le pregunta si desea eliminarlo o no (cuando este se está editando y no selecciona ningún campo para modificar).
  2. Una vez se tome la decisión, este se borra o no de lo que se almacene en la memoria temporal, se sale y guarda el cambio en la memoria permanente.
- Requerimientos funcionales:
  - Se consulta por medio de la consola si el usuario va a ser o no eliminado.
  - La respuesta únicamente debe ser "Y", de lo contrario, se asume que el usuario no desea ser eliminado.
  - Una vez se borre, se debe cerrar con éxito la página principal, y guardar el cambio en la memoria permanente.

F. Lectura de usuarios [Read/Leer]

- Descripción: Se leen todos los datos en el archivo JSON correspondiente a los usuarios y se carga en la memoria temporal.
- Acciones iniciadoras y comportamiento esperado:
  1. La acción inicializadora de esta funcionalidad es el mismo mensaje de bienvenida, ya que

posteriormente se utiliza para la comprobación, creación y eliminación de cada dato.
2. Se carga la información a la memoria temporal.

- Requerimientos funcionales:
  - Se requiere de una librería que abra correctamente los archivos JSON, y de espacio suficiente en disco para poder almacenar la información.
  - Como la acción se ejecuta únicamente con el parámetro de la ruta, en caso de no encontrarlo, se debe crear un archivo vacío sobre el que escriba más adelante.

G. Almacenamiento en memoria permanente [Creación]

- Descripción: Se guardan los campos de manera permanente en archivos de extensión JSON.
- Acciones iniciadoras y comportamiento esperado:
  1. La acción se ejecuta una vez el usuario desee guardar y salir, esto se pregunta cada vez que se le dirija a la página principal.
  2. Se sobre escribe el archivo que fue abierto o creado en el momento de hacer la lectura. *Véase requerimientos funcionales (F).*
  3. Una vez se guarde en memoria permanente, el programa finaliza su ejecución.
- Requerimientos funcionales:
  - Se requiere de una librería que abra correctamente los archivos JSON, y de espacio suficiente en disco para poder almacenar la información.
  - Como la acción se ejecuta únicamente con el parámetro de la ruta, en caso de no encontrarlo, se debe crear un archivo vacío sobre el que escriba.

H. Creación de proyectos [Creación y lectura]

- Descripción: Se crean los proyectos que estén asociados a la cuenta que está iniciada.
- Acciones iniciadoras y comportamiento esperado:
  1. La acción se ejecuta cada vez que el usuario desee crear un nuevo proyecto. Esto se sabe porque se le pregunta y debe responder de manera afirmativa con "Y".
  2. Una vez se escoja la opción, se solicita cambiar los datos correspondientes mas fundamentales (Nombre del proyecto y Presupuesto).
  3. Una vez finalice el proceso se le muestra un resumen al usuario de los proyectos que ha creado por el momento con un resumen de los usuarios que los siguen.
- Requerimientos funcionales:
  - Se debe insertar de manera obligatoria una entrada numérica para el presupuesto, de lo contrario se debe desplegar un mensaje de error.
  - Los mensajes de entrada y salida se realizan a través de consola.
  - Los Nombres y resumen de los proyectos se muestran tras hacer el recorrido de la lista asociada de proyectos que tiene el usuario.

I. Edición de Proyectos [Edición y lectura]

- Descripción: Se Solicita una vez se cree l proyecto si se desean hacer cambios de inmediato, o cuando el proyecto se busque en otra instancia.
- Acciones iniciadoras y comportamiento esperado:
  1. Una vez creado el proyecto, puede ser en el instante o cuando se consulte si este desea editar o no el proyecto. Atributos como lo son el nombre y el presupuesto estimado.
  2. Una vez tomada la decisión, el usuario deberá elegir que datos son los que desea modificar.
  3. Una vez finalice el proceso se pasa a la pagina que estaba inmediatamente antes que esta última.
- Requerimientos funcionales:
  - Se debe insertar de manera obligatoria una entrada numérica para el presupuesto, de lo contrario se debe desplegar un mensaje de error.
  - Los mensajes de entrada y salida se realizan a través de consola.
  - Los Nombres y resumen de los proyectos se muestran tras hacer el recorrido de la lista asociada de proyectos que tiene el usuario.
  - La navegación para saber la instancia inmediatamente anterior se realiza con el uso de pilas (Forward y Back Ward) Su uso arbitrario por el usuario deberá ser implementado en otra ocasión.

J. Eliminación de proyectos [Eliminación]

- Descripción: Si el usuario no decide que campo editar, se muestra la opción de eliminar el proyecto.
- Acciones iniciadoras y comportamiento esperado:
  1. Una vez seleccionado el proyecto que se desea eliminar y se han rechazado las posibilidades de editar alguno de sus campos, el usuario deberá elegir si lo elimina o no.
  2. El objeto se elimina de las listas asociadas tanto al usuario como de los proyectos existentes y el cambio será permanente cuando se salga de la sesión y se guarde. *Véase Requerimientos funcionales (G).*
- Requerimientos funcionales:
  - Se debe insertar de manera obligatoria la opción "Y", de lo contrario se asume que el usuario no desea eliminar dicho proyecto.
  - Los mensajes de entrada y salida se realizan a través de consola.
  - La eliminación se hace en las listas asociadas al usuario y a los proyectos existentes.

V. Descripción de la interfaz de usuario preliminar

La interfaz de usuario consistirá en la elaboración de una aplicación de escritorio (por el momento), el diseño preliminar de la solución se pudo definir con el uso de mockups, utilizando una herramienta llamada "*Wireframe Pro*" [4]. Los Mockups diseñados son los que se muestran a continuación:

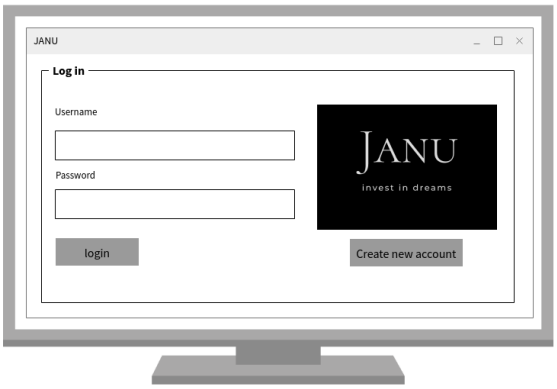


Ilustración 1: Login

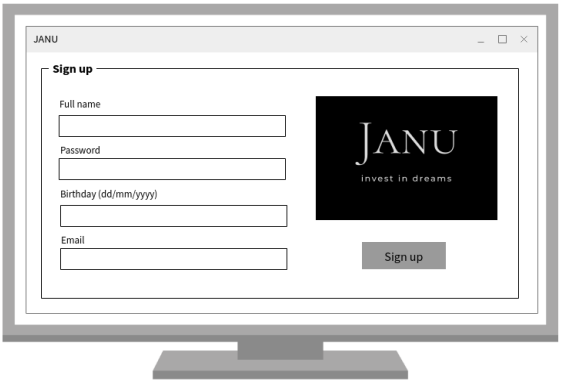


Ilustración 2 Registro (Signup)

Ilustración 3 Funcionamiento principal

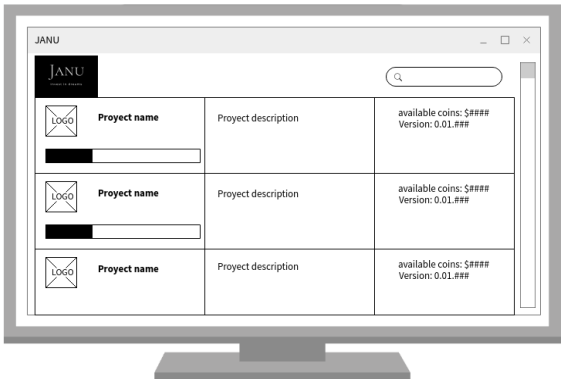


Ilustración 4 Buscador de proyectos

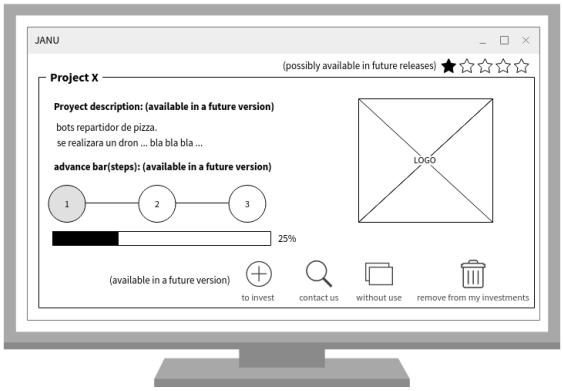


Ilustración 5 Vista de proyectos

VI. Entornos de desarrollo y de operación

- Editor y entorno de desarrollo: Visual Studio Code (Con plugins de Java)
- Memoria temporal (RAM): 16 GB @ 2666 MHz
- Procesador: intel Core i7 (8750H) @ 3.9 GHz
- Memoria permanente: 512 GB SSD

VII. Prototipo de software inicial

El progreso del prototipo con las funcionalidades definidas anteriormente como primer avance se encuentran en el repositorio referenciado en el siguiente enlace: https://github.com/dasilvaca/Data-Structures-2020

VIII. Implementación y aplicación de las estructuras de datos

Para esta primera etapa del proyecto se usaron las siguientes estructuras de datos secuenciales:

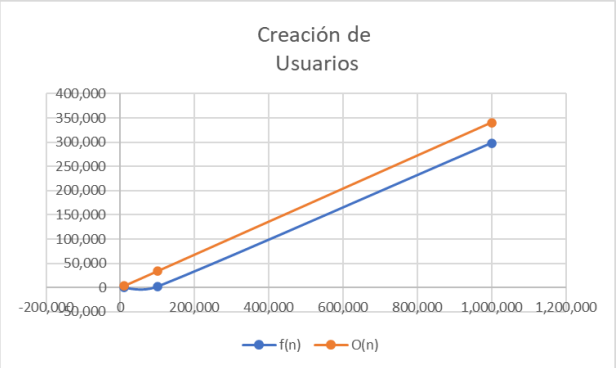
- Arreglos Dinámicos: En ellas se almacenan en memoria temporal tanto a los usuarios como a los proyectos asociados.
- Listas enlazadas: Se guardan en ella los usuarios asociados como dueños a cada proyecto.
- Pilas: Se utilizan para la navegación en ejecución para retroceder y avanzar (Con esta se conoce la instancia de la ventana inmediatamente anterior)
- Colas: No se implementaron en esta ocasión las colas, de ser necesario se hará en futuras versiones (Sin embargo se creó dentro del proyecto).

IX. Pruebas del prototipo y análisis comparativo

Se realizaron pruebas para cada operación por cada tanto número de datos en tantos milisegundos:

- Creación de usuarios:
  - 10.000: 31
  - 100.000: 1.763
  - 1’000.000: 298.262
  - 10’000.000: + 8 horas (Java heap space)
  - 100’000.000: + 8 horas (Java heap space)

Cantidad de datos	Tiempo (ms)	O(n)
10,000	31	3400
100,000	1,763	34000
1,000,000	298,262	340000



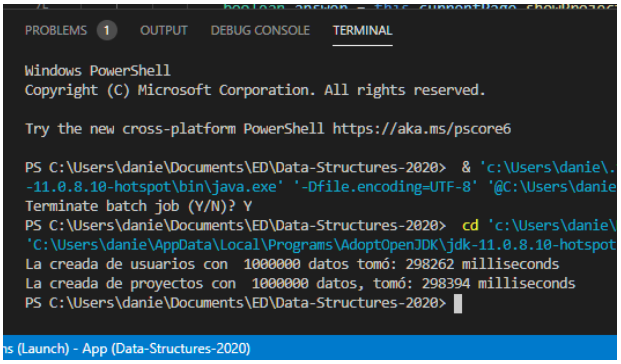
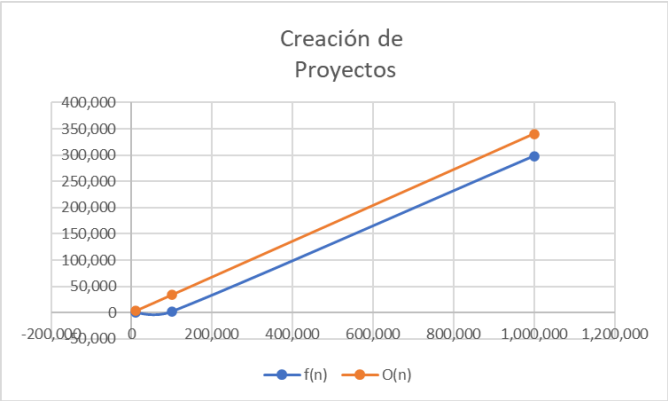


Ilustración 6: Prueba creación de Usuarios y Proeyctos (1 millón de datos)

- Creación de proyectos:
  - 10.000: 18
  - 100.000: 1.438
  - 1’000.000: 298.394
  - 10’000.000: + 8 horas (Java heap space)
  - 100’000.000: + 8 horas (Java heap space)

Cantidad de datos	Tiempo (ms)	O(n)
10,000	18	3400
100,000	1,438	34000
1,000,000	298,394	340000



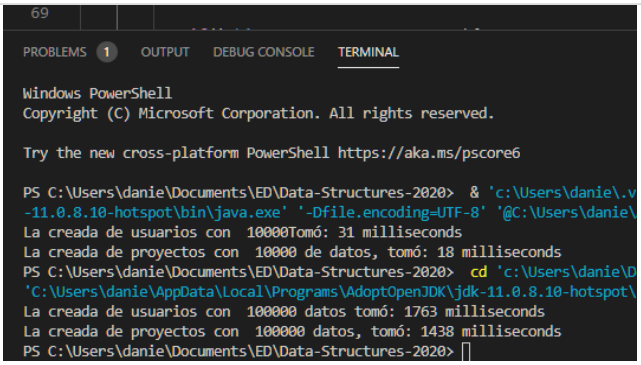


Ilustración 7 : Prueba creación de usuarios y proyectos 100 mil datos

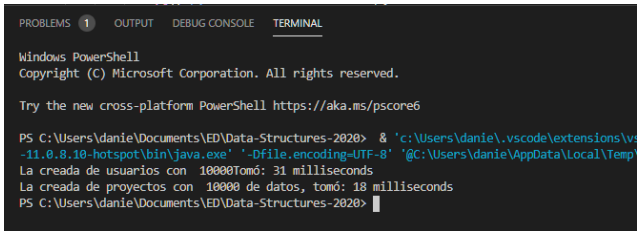
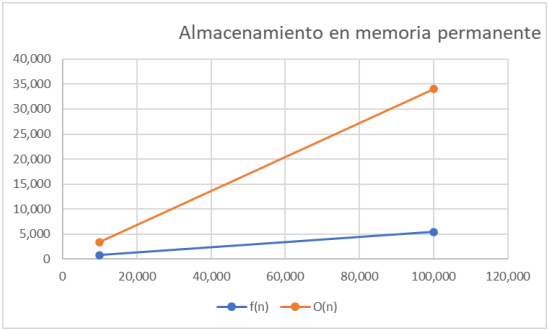


Ilustración 8 Prueba creación de Usuarios y proyectos (10 mil datos)

- Escritura en memoria permanente:
  - 10.000: 838
  - 100.000: 5.486
  - 1’000.000: Memoria insuficiente (JSON no lo soportó)
  - 10’000.000: Memoria insuficiente (JSON no lo soportó)
  - 100’000.000: Memoria insuficiente (JSON no lo soportó)

Cantidad de datos	Tiempo (ms)	O(n)
10,000	838	3400
100,000	5,486	34000



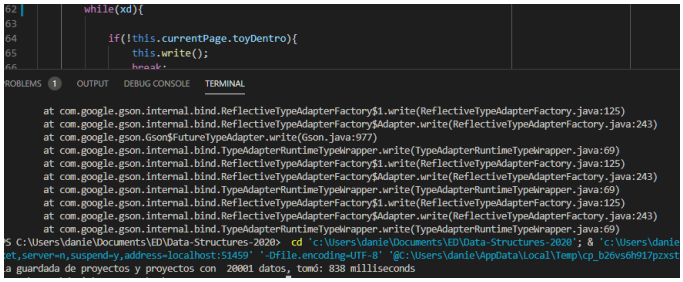


Ilustración 9 Prueba almacenamiento en memoria permanente 20.000 datos (1000 Usuarios con un proyecto c/u)

Ilustración 10 Prueba almacenamiento permanente con más de un millón de datos (Límite de memoria alcanzado)

A. Análisis:

Dado que las representaciones hechas se pueden acotar superiormente con una función constante (En cada caso se usó la pendiente 0.34) decimos que los procedimientos son de orden O(n), sin embargo, como muchos de los casos no se pudieron estudiar debido al tiempo, no podemos afirmarlo con certeza, ya que en un momento dado debería de culminar, o es la hipótesis que plantea el equipo de trabajo.

X. Roles y actividades

	Actividades fundamentales
Líder/esa	Silva Capera Daniel Santiago
Coordinador/a:	Bedoya Torres Juan Eduardo, Moreno Hernández José Luis, Silva Capera Daniel Santiago
Experto:	Bedoya Torres Juan Eduardo, Moreno Hernández José Luis, Silva Capera Daniel Santiago
Investigador	Moreno Hernández José Luis, González Castro Camilo Andrés
Observador	Silva Capera Daniel Santiago
Animador/a	Bedoya Torres Juan Eduardo, Moreno Hernández José Luis, Silva Capera Daniel Santiago
Secretario	Bedoya Torres Juan Eduardo, Moreno Hernández José Luis, Silva Capera Daniel Santiago
Técnico	Bedoya Torres Juan Eduardo, Moreno Hernández José Luis, Silva Capera Daniel Santiago

INTEGRANTE	**ROL**	**ACTIVIDADES**
Bedoya Torres Juan Eduardo	Animador	Interfaces de usuario y Navegación
	Secretario	Stacks/ Pilas
	Técnico	Clase Proyecto
	Coordinador	Clase Usuario
	Experto	Planeación
		Nodos
Moreno Hernández José Luis	Animador	Priority Queues
	Secretario	Listas enlazadas
	Técnico	JSON (Lectura y Escritura)
	Coordinador	Planeación
	Experto	Lista doblemente enlazada
	Investigador	Arboles (Por implementar)
González Castro Camilo Andrés	Investigador	Diapositivas
		Mockups
		Queues/ Colas
Silva Capera Daniel Santiago	Observador	Priority Queues
	Animador	Dynamic array
	Secretario	Enseñar a usar Git y VS code
	Técnico	Planeación
	Líder/esa	Organización de directorios y Paquetes
	Coordinador	Coordinador de reuniones
	Experto	Trabajo escrito
		Pruebas (Análisis)

XI. Dificultades y lecciones aprendidas

Se experimentaron problemas de comunicación con algunos de los miembros (en su minoría). No se logró finalizar el proyecto a tiempo pese a diferentes situaciones como errores y fallas con el sistema de control de versiones utilizado. La transmisión del conocimiento llegó a ser tardía en el caso de la lectura y escritura de archivos. Se presentaron inconvenientes de tiempo, aunque siempre la mayoría estuvo dispuesta a trabajar. Como lección, se aprendió que siempre es mejor un buen diseño desde el principio, y que en ocasiones la improvisación y creatividad en el camino puede resultar difícil de controlar o implementar. Además al ser un desarrollo relativamente grande debido a la experiencia de los integrantes, realmente fue sorpresivo y preocupante el ver como cada vez que se pensaba en ello podía transformarse en un desarrollo más grande, por ende es mejor escalar los proyectos e implementar funcionalidades menores desde el principio. También se debe mejorar en la organización y optimización del código para hacerlo más legible. Se subestimó el tiempo que podía llegar a tomar la muestra de los datos para cada uno de los volúmenes.

XII. Referencias

[1] El Espectador, «Desempeño de la economía colombiana en 2020,» [En línea]. Available: https://bit.ly/3he35rH.

[2] Wikipedia, «Kickstarter,» [En línea]. Available: "https://en.wikipedia.org/wiki/Kickstarter.

[3] Kickstarter, «Manual del creador,» [En línea]. Available: https://www.kickstarter.com/rules?ref=global-footer.

[4] Kickstarter, «Kickstarter Colombia,» [En línea]. Available: https://bit.ly/2R21vPh