Título del Proyecto

Primer Autor, Segundo Autor, Tercer Autor

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

I. [1] INTRODUCCIÓN

Este documento contiene la información del estado del proyecto “Stats Tracker”, de la misma forma, se plantean ideas para la versión final de la aplicación. Y toda la información referente al proceso del desarrollo de la aplicación, y como su desarrollo puede verse afectado.

I. DESCRIPCIÓN DEL PROBLEMA

Se busca responder a la interrogante ¿Es posible crear una aplicación que le permita al usuario registrar sus estadísticas para tener un posterior seguimiento de estas? Entre las estadísticas se puede encontrar, puntuaciones, muertes, desempeño en general, de diferentes videojuegos. Mediante una interfaz sencilla de comprender para un usuario promedio.

Actualmente hay diferentes soluciones a esto en el mercado, por ejemplo “Tracker Network” o “Blitz” para determinados juegos. Otras aplicaciones, como “Discord”, que informan cuánto tiempo lleva en determinada sesión de juego. Sin embargo, no existe una aplicación que integre todas estas soluciones en una misma, para diferentes videojuegos.

II. USUARIOS DEL PRODUCTO DE SOFTWARE

Esta aplicación tiene como público objetivo personas que juegan videojuegos, se diseña para que sea de fácil entendimiento tanto para niños, adolescentes y adultos, sin embargo, se espera un mayor uso de esta aplicación por parte de personas que se encuentran entre la adolescencia y la adultez.

III. REQUERIMIENTOS FUNCIONALES DEL SOFTWARE

Para obtener los resultados planteados en la aplicación, es necesario que cumpla con determinados requisitos de software.
Estos requisitos se presentan como una suma de los requisitos esperados para esta entrega, en conjunto de los requerimientos de la entrega pasada.
En primer lugar, se necesita iniciar sesión o registrarse, debido a que esta aplicación debe de ser compatible con varios juegos en diferentes dispositivos, además del soporte con varios juegos de forma nativa.

- *Creación de usuarios*

- *Descripción:*

Permite a los usuarios crear una cuenta en la cual tener sus estadísticas guardadas para editarlas o crear nuevas.

- *Acciones iniciadoras y comportamiento esperado*:

Recién inicia la aplicación, el usuario deberá iniciar sesión o crear una cuenta desde el primer menú, después de un inicio de sesión exitoso, lo dirigirá a la siguiente pantalla. Cuando se crea un usuario, también se le asignará un hash único que depende de su nombre.

Requerimientos funcionales:

- Obtención de datos del juego (Soporte nativo).

- *Descripción:*

Permite acceder a los datos de un juego en específico que tenga soporte el juego, tales como Celeste, es decir, obtiene los datos para las estadísticas sin necesidad de que el usuario los suministre.

- *Acciones iniciadoras y comportamiento esperado*:

Al seleccionar un juego que tenga soporte con la aplicación, se debe reconocer como unos de los juegos con soporte de la aplicación, por ende, no necesitará que el usuario administre datos para mostrar la gráfica de su rendimiento.

Requerimientos funcionales:

- Obtención de datos del juego (Sin soporte nativo).

- *Descripción:*

Cuando un juego no tiene soporte nativo para la aplicación, el usuario debe proporcionar los datos que desea seguir con la aplicación.

- *Acciones iniciadoras y comportamiento esperado*:

Al seleccionar un juego que no tenga soporte con la aplicación, el usuario debe agregar sus propios datos, que serán subidos posteriormente a la base de datos.
.

[1]

Requerimientos funcionales:

- Consulta de datos.

- *Descripción:*

Este requerimiento le pide a la aplicación los datos, para realizar la gráfica correspondiente al rendimiento.

- *Acciones iniciadoras y comportamiento esperado*:

Si el juego tiene soporte nativo para la aplicación, esta misma tomará los datos propios de la aplicación, que serán almacenados por el programa. Si no tiene soporte nativo para la aplicación, el programa deberá acceder a la base de datos, y consultar con respecto a la id y al juego seleccionados y anteriormente dados.

Requerimientos funcionales:

- Ordenamiento de datos (Cuando se tiene soporte nativo de la app).

- *Descripción:*

Este requerimiento tomará los datos tomados de la aplicación, y los organizará para su posterior gráfica.

- *Acciones iniciadoras y comportamiento esperado*:

Si el juego tiene soporte nativo para la aplicación, y el usuario selecciona estadísticas para esta aplicación, se acomodará en árboles para su fácil búsqueda.

Requerimientos funcionales:

- Impresión de estadísticas.

- *Descripción:*

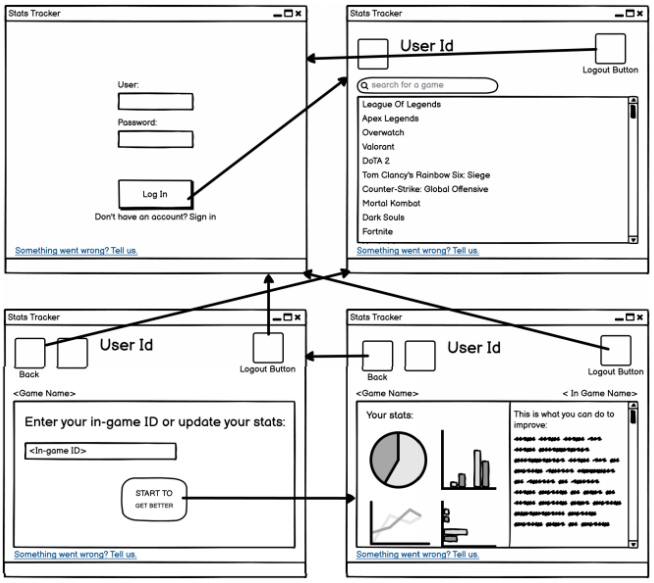
Con los datos recopilados y acomodados, imprimirá estos datos en una gráfica, que mostrará .

- *Acciones iniciadoras y comportamiento esperado*:

Después de seleccionar el juego o actualizar los datos, dependiendo el caso, la aplicación va a mostrarlos datos basados con los datos estadísticos de las últimas 10 actualizaciones.

IV. AVANCE EN LA IMPLEMENTACIÓN DE LA INTERFAZ DE USUARIO

La interfaz de usuario continua como se ha planteado desde la primera vez que se propuso la interfaz en la primera entrega, con cuatro diferentes menús, que intuitivamente llevan al usuario a encontrar sus datos, con pocos botones, sin embargo, por las limitaciones de la librería, no es posible tener una interfaz tan limpia como se desea.

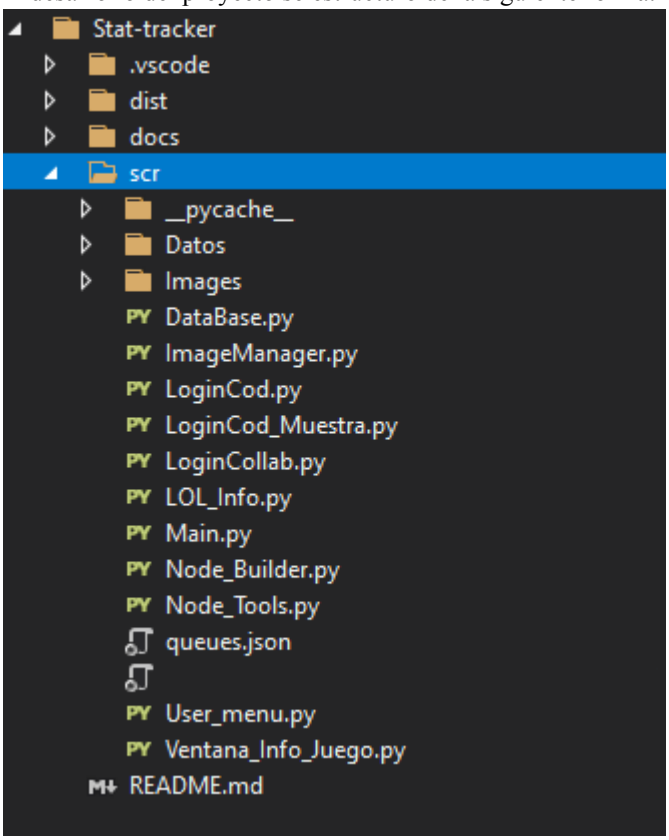


V. ENTORNOS DE DESARROLLO Y DE OPERACIÓN

Para el desarrollo de este software, se utilizó el lenguaje de programación Python y varias librerías que este ofrece, se idea la aplicación para funcionar en Windows 10, sin embargo, no se espera que el hardware limite de ninguna forma la aplicación (Más allá de la conexión a internet).

VI. DESCRIPCIÓN DEL PROTOTIPO DE SOFTWARE

El desarrollo del proyecto se estructuró de la siguiente forma:



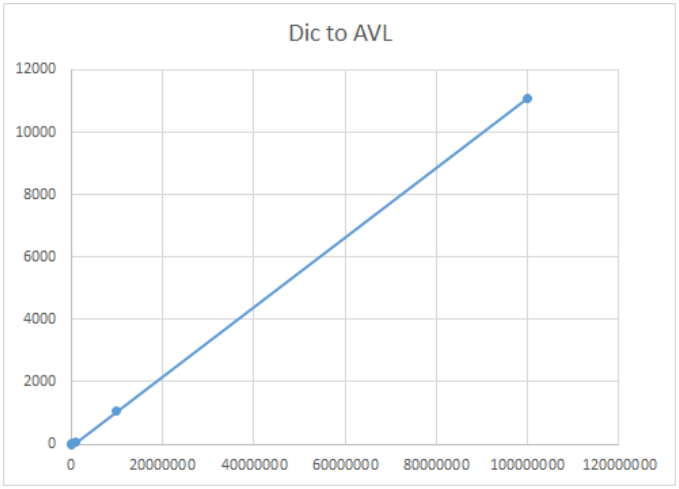
Sobre las estructuras de datos utilizadas en el prototipo, se tiene una Lista Enlazada para mostrar el historial de partidas de League of Legends (Uno de los juegos que soporte la aplicación). También se utilizaron Arboles AVL para el ordenamiento de los datos para los juegos que tienen soporte nativo en la aplicación. Adicionalmente se adiciona la librería de hash, propia de Python, para poder encriptar los id de los usuarios y referenciarlos así en la base de datos.

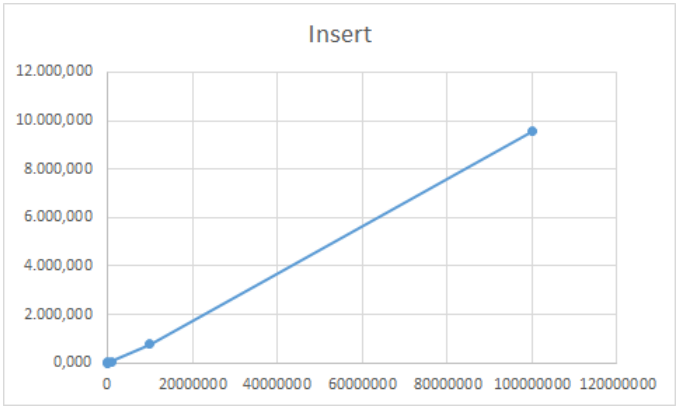
VII. PRUEBAS DEL PROTOTIPO

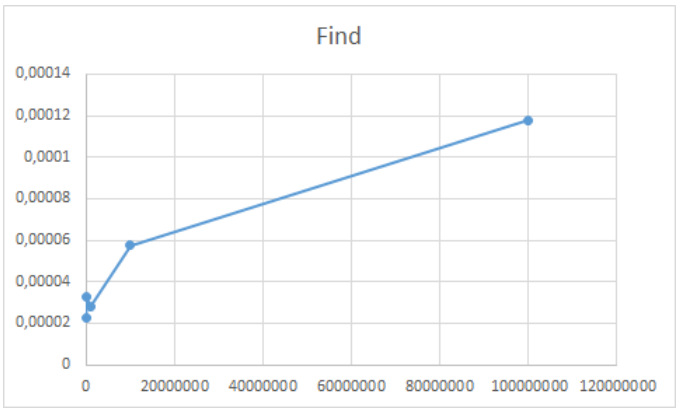
Tabla 1. Tabla comparativa de las pruebas realizadas.

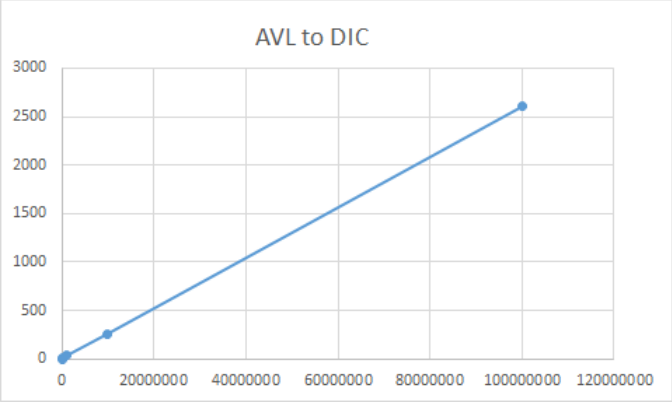
Nombre de la funcionalidad (Camilo Fierro)	Tipo(s) de estructura de datos	Cantidad de datos probados	Tiempos de ejecución (segundos)	Análisis realizado (Notación Big O)
Dictionary to AVL	Árbol AVL y diccionario	10000	0.0005	O((n))
		100000	4.3333	
		1000000	52.972	
		10000000	1061.101	
		100000000*	11106.7	
Insert()	Árbol AVL	10000	0.410	O(n)
		100000	4.758	
		1000000	59.948	
		10000000	757.361	
		100000000*	9568.22052	
Find()	Árbol AVL	10000	2.24e-05	O(log(n))
		100000	3.27e-05	
		1000000	2.80e-05	
		10000000	5.75e-05	
		100000000*	1.18e-04	
AVL to dictionary	Árbol AVL y diccionario	10000	0.186	O(n)
		100000	3.325	
		1000000	33.466	
		10000000	257.4962	
		100000000*	1981.243	

*Para esta tabla, la máquina del usuario no fue suficiente y debido a esto la información se tomó como un supuesto en base al crecimiento de las pruebas anteriores.









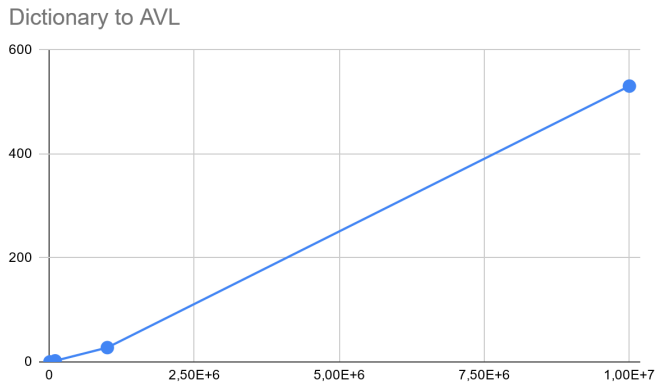
Para este caso, a pesar de que la máquina no fue capaz de realizar correctamente las pruebas, si se debe destacar que fue supremamente más eficiente en comparación con las listas y colas que se usaban anteriormente, lo que demuestra que la eficiencia si se logró pero el orden debe verse afectado por algún otro factor dentro del código. Y esto también se ve en el caso del find que si llega a ser O(log(n)) pues solo se comunica con el árbol, a diferencia del resto de funciones que trabajan con árboles y diccionarios a la vez.

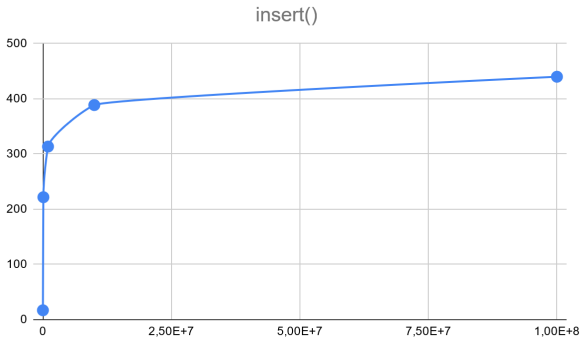
Tabla 2.

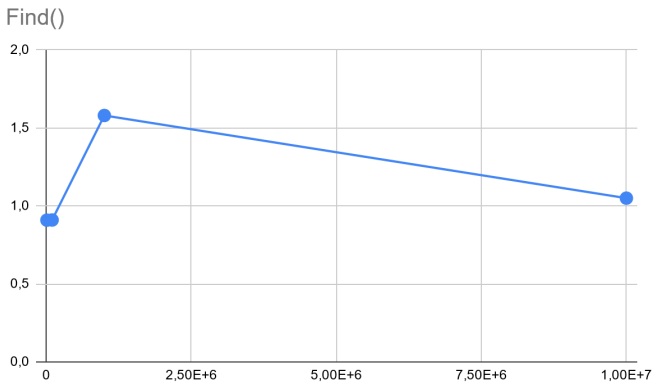
Nombre de la funcionalidad (Baruj)	Tipo(s) de estructura de datos	Cantidad de datos probados	Tiempos de ejecución (segundos)	Análisis realizado (Notación Big O)
Dictionary to AVL	Árbol AVL y diccionario	10000	0.17701	O(log(n))
		100000	2.14779	
		1000000	27.5560	
		10000000	530,55*	
		100000000	-	
Insert()	Árbol AVL	10000	0.16967	O(log(n))
		100000	2.21810	
		1000000	31.3515	
		10000000	388.701	
		100000000	440*	
Find()	Árbol AVL	10000	9.09e-06	O(log(n))
		100000	9.1e-06	
		1000000	1.58e-05	
		10000000	1,05e-05*	
		100000000	2,11e-05*	
AVL to dictionary	Árbol AVL y diccionario	10000	0.18544	O(log(n))
		100000	1.87808	
		1000000	18.9664	
		10000000	128,74*	
		100000000	-	

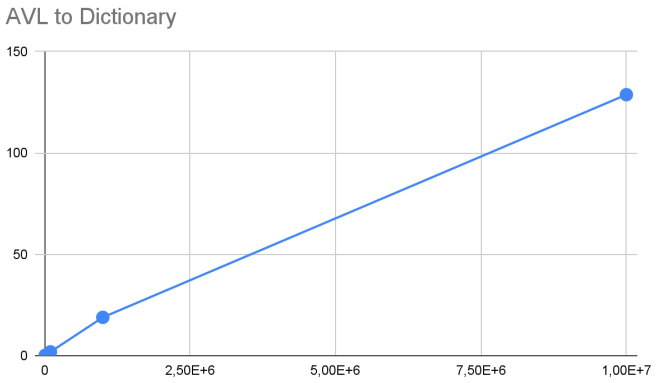
*Los valores son aproximaciones basándose en los otros datos obtenidos

-Algunos datos no se pudieron aproximar correctamente y por esta razón se prefirió excluirlos para tener una mejor idea general con el resto de datos.









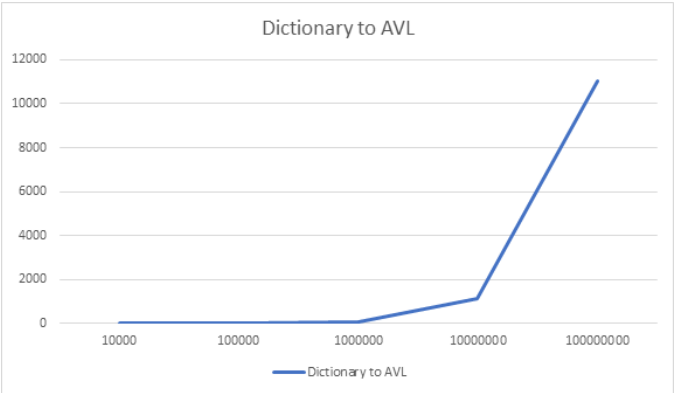
Se puede apreciar en insert(), dictionary to AVL y AVL to dictionary que corresponden a su análisis de notación Big O, en los datos obtenidos de Find() no se puede apreciar, pues el tiempo que tomó en recorrer un árbol con n datos fue bastante pequeño, lo suficiente para que los tiempos obtenidos se vieran influenciados por factores internos en el computador no necesariamente relacionados con la cantidad de datos.

Tabla 3.

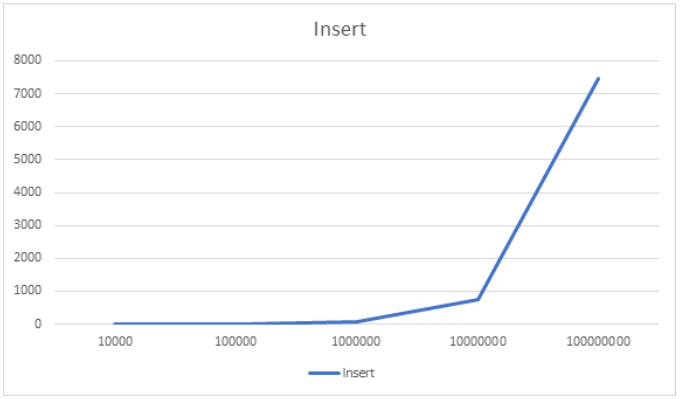
Nombre de la funcionalidad (Fabian Rozo)	Tipo(s) de estructuras de datos	Cantidad de datos probados	Tiempos de ejecución (segundos)	Análisis realizado (Notación Big O)
Dictionary to AVL	Árbol AVL y diccionario	10000	0.012	O(n)
		100000	4.153	
		1000000	54.19	
		10000000	1104.2	
		100000000	11042*	
Insert()	Árbol AVL	10000	0.327	O(log(n))
		100000	4.157	
		1000000	60.7	
		10000000	748.23	
		100000000	1000	
Find()	Árbol AVL	10000	1.28e-05	O(log(n)
		100000	1.27e-05	
		1000000	1.7e-05	
		10000000	2.11e-05	
		100000000	4.22e-05 *	
AVL to Dictionary	Árbol AVL y diccionario	10000	0.15	O(Log(n))
		100000	3.001	
		1000000	30.95	
		10000000	250.84	
		100000000	2508.4*	

Los valores que tienen asterisco (*), no pudieron ser comprobados mediante pruebas de software, por limitaciones.

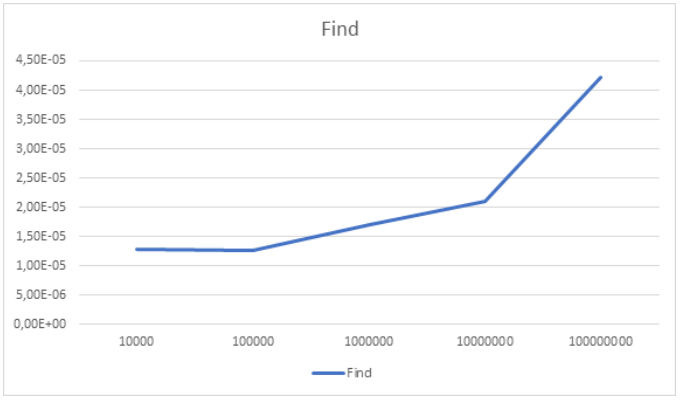
Dictionary to AVL



Insert



Find



AVL to Dictionary

Se enfatiza que, además de las tablas comparativas, se deben presentar gráficas que ilustran el análisis de complejidad realizado.

VIII. ACCESO AL VIDEO DEMOSTRATIVO DE SOFTWARE

https://youtu.be/ON1VUgmnRNc

IX. ROLES Y ACTIVIDADES

INTEGRANTE	ROL(ES)	ACTIVIDADES REALIZADAS (Listado)
Camilo Andres Fierro Fierro	Técnico	Idear las funciones y el cómo estas se conectan con la interfaz
		Ayudar con el desarrollo del modelo de los árboles AVL
	Experto	Introducir la herramienta con la que se desarrolla la interfaz gráfica
Baruj Vladimir Ramirez Escalante	coordinador	Facilitar la comunicación del grupo por medio de la plataforma de Whatsapp y acordar tiempos para las reuniones
	Líder	Facilitar la distribución de tareas del equipo.
Fabian Andres Rozo Rodriguez	Investigador	Investigar sobre formas de agregar el hash en el código
		Encontrar la alternativa firebase, una base de datos que usamos para almacenar los archivos
	Observador	Observar atentamente al desarrollo de la aplicación
		Realizar un backup local de la aplicación, por si algo funciona mal

X. DIFICULTADES Y LECCIONES APRENDIDAS

A lo largo del desarrollo del proyecto hemos notado como el desarrollo de una idea que en principio resulta simple se puede volver una tarea complicada cuando se ejecuta y transforma repetidamente. Las estructuras de datos ayudan mucho a la mejora de la eficiencia cuando la carga de datos es masiva pero la teoría que tiene detrás es compleja en muchos aspectos y la aplicación de la misma no resulta de la manera en la que se espera. Durante el desarrollo de los árboles surgieron

numerosos problemas acompañados de la incertidumbre de no saber si el error era nuestro al escribir mal el código o de la teoría al estar aplicándola donde no se debía. El trabajo en equipo que ha surgido de este proyecto nos ha enseñado que a pesar de que cada persona tiene sus circunstancias, al momento de trabajar esto hace que las ideas se complementan y donde uno se atora y no encuentra solución, el equipo puede desenvolverse y solucionar el dilema desde otro punto de vista.

La falta de experiencia a la hora de hacer una interfaz gráfica nativa en Python produjo que el avance se viera ralentizado considerablemente, optando por posponer la interfaz gráfica para implementar más funcionalidades.

XI. REFERENCIAS BIBLIOGRÁFICAS

(2019, September 20) Análisis asintótico [Online]. Available: https://es.wikipedia.org/wiki/Análisis_asintótico

H. Barney (2020, March 20)How To Use Riot API With Python[Online]. Available: https://towardsdatascience.com/how-to-use-riot-api-with-python-b93be82dbbd6

LEAGUE OF LEGENDS [Online]. Available: https://developer.riotgames.com/docs/lol

J.Quick (2019,December 12) How To use Live Share with Visual Studio Code[Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-use-live-share-with-visual-studio-code#:~:text=Live%20Share%20is%20an%20extension,a%20sever%20and%20debugging%20session.