Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

Algoritmos y Estructuras de Datos II

Proyecto #3

Trabajo de Investigación

Tema: My Invincible Library

Profesor:
Luis Diego Noguera Mena

Estudiantes:
Saymon Astúa Madrigal, 2018143188
Kevin Acevedo Rodríguez, 2018148661
Kennet Mora Prado, 2018072063
Oscar Araya Garbanzo, 2018002998

Cartago, 18 de junio 2019

Tabla de Contenido

Introducción

My Invencible Library se trata de un proyecto que permite el almacenamiento de galerías de imágenes con un sistema particular de almacenamiento a de galerías de imágenes con posibilidad de reponerse a fallos, esta se conforma de cuatro elementos:

- ServerLibrary: Permite a múltiples clientes acceder a la información de MyInvincibleLibrary.
- RAID Library: permite el almacenamiento de las galerías y brinda redundancia al sistema.
- Metadata DB: base de datos NoSQL para el almacenamiento de la metadata de las imágenes.
- MyIDE: permite a los múltiples clientes gestionar la metadata de las galerías de imágenes.

Donde una galería está conformada por un conjunto de imágenes. El sistema puede estar conformado por un conjunto de galerías.

Se espera que el sistema permite realizar todas las operaciones desde MyIDE y que en caso de realizar consultas se puedan abrir las imágenes asociadas a los resultados mostrados, aún cuando se pierda alguno de los discos virtuales que almacenan la imagen.

Análisis

El sistema cuenta con los siguientes elementos:

RAID Library:

Se debe implementar una biblioteca que funcione como controlador de RAID 5. Esta biblioteca debe implementar al menos las funciones:

1. Read (capacidad de lectura).
2. Write (capacidad de escritura).
3. Seek (capacidad de búsqueda).

Dichas funciones deben comportarse de manera similar a las funciones tradicionales para leer/escribir/desplazarse en archivos. El controlador se encarga de gestionar los discos virtuales. Cada disco virtual es una única carpeta en el sistema de archivos donde se encuentran los bloques de cada archivo.

Por ejemplo, suponga que se guarda una imagen de n bytes. El controlador recibe la llamada a write, divide el archivo en bloques y para cada bloque crea un archivo de los discos virtuales. Adicionalmente calcula el bloque de paridad.

El controlador debe mantener cualquier información adicional para poder encontrar cada uno de los bloques del archivo y para poder recuperar datos en caso de perder alguno de los discos virtuales

Metadata DB:

Es una base de datos NoSQL. La información debe almacenarse en archivos JSON o XML. Cada grupo debe definir la estructura de los archivos y la base de datos como tal. Para las imágenes es necesario, como mínimo, gestionar la siguiente información:

1. Nombre de la imagen.
2. Autor.
3. Año de creación.
4. Tamaño.
5. Descripción.

Server Library:

Permite la comunicación entre los múltiples clientes y My Invencible Library. Queda a criterio de cada grupo la implementación de este servidor. Pueden considerar hacerlo mediante sockets, REST API o algún otro que se considere oportuno.

My IDE:

Consiste en una aplicación que se instala en el cliente y sirve para poder acceder al servicio de Invencible. Debe ser implementado en C++ para Linux y además debe permitir las siguientes operaciones:

1. Agregar metadata.
2. Consultar metadata.
3. Eliminar metadata (al eliminar la metadata debe eliminarse también la imagen asociada a esta

metadata).

4. Modificar metadata.

Estas cuatro operaciones deben realizar siguiendo la sintaxis de SQL (INSERT, SELECT, DELETE y UPDATE). Las consultas en el IDE, además de devolver la metadata seleccionada debe agregar una opción para abrir la imagen asociada a la metadata que se despliegue después de ejecutar la consulta. Adicionalmente el IDE debe proveer la posibilidad de agregar galerías o imágenes. Esta funcionalidad no debe ser implementada mediante scripts.

esto en la planificación del proyecto se dividió en lo siguiente:

- Implementar Comunicación cliente-servidor
- Almacenar la información de la metadata en formato JSON dentro de metadata DB
- Crear una base de datos NoSQL no relacional para la Metadata DB
- Implementar un controlador que se encargue de manejar el RAID 5
- Crear una interfaz para interactuar con los elementos del servidor
- Activar función de write desde el controlador
- Habilitar la utilización de los operadores básicos de SQL dentro de My IDE
- Gestionar las carpetas lógicas desde el controlador RAID 5
- Simular el funcionamiento de un RAID 5
- envío de datos JSON
- Simular el funcionamiento de una galería, que contiene distintos álbumes de imágenes
- Almacenar parte de la imagen en distintas carpetas del RAID 5
- Simular desde el RAID 5 el funcionamiento de un manejador de archivos tradicional.

También realizado en la parte de diseño se puede obtener requerimientos como los siguientes :

- Yo como usuario quiero que MyInvencibleLibrary funcione como un sistema cliente/servidor.
- Yo como usuario quiero poder almacenar galerías de imágenes.
- Yo como usuario quiero que el sistema tenga la capacidad de reponerse a fallos.
- Yo como usuario quiero que el sistema cuente con ServerLibrary, RAID Library, Metadata DB y MyIDE.
- Yo como usuario quiero que se puedan mostrar las imágenes desde MyIDE, aun cuando se pierda alguno de los discos virtuales que almacenan esta imagen.
- Yo como usuario quiero que se puedan agregar galerías o imágenes desde MyIDE (no por medio de scripts).
- Yo como usuario quiero que se puedan abrir las imágenes asociadas a alguna metadata desde MyIDE.
- Yo como usuario quiero que esta operaciones de MyIDE sigan la sintaxis de SQL (INSERT, SELECT, UPDATE Y DELETE).
- Yo como usuario quiero que MyIDE permita agregar metadata, consultar metadata, elminar metadata y modificar metadata.

- Yo como usuario quiero que MyIDE sea una aplicación que se instale en los cliente, programada en C++ para el sistema operativo Linux.

- Yo como usuario quiero que para cada imágen en el Metadata DB gestione la información de nombre, autor, año de creación, tamaño y descripción.

- Yo como usuario quiero que la metadata de Metadata DB se almacene en archivos JSON o XML.

- Yo como usuario quiero que el sistema de Metadata DB sea una base de datos NoSQL.

- Yo como usuario quiero que el controlador mantenga información adicional para encontrar los bloques del archivo y para poder recuperar datos en caso de perder alguno de los discos virtuales.

- Yo como usuario quiero que cada vez que se almacene una imagen de n bytes se calcule el bit de paridad para cada bloque de información.

- Yo como usuario quiero que para cada uno de estos bloques se cree un archivo en las carpetas de los discos virtuales.

- Yo como usuario quiero que si se quiere guardar un imagen de n bytes el controlador active la función de escritura (write) y se divida el archivo en bloques.

- Yo como usuario quiero que cada uno de estos discos virtuales sea una única carpeta en el sistema de archivos donde se encuentran los bloques de cada archivo.

- Yo como usuario quiero que el controlador del RAID Library gestione los discos virtuales.

- Yo como usuario quiero que estas funciones se comporten como las funciones tradicionales de manejo de archivos.

- Yo como usuario quiero que se implemente una biblioteca que funcione como controlador de RAID 5 dentro del RAID Library, que sea capaz de leer, escribir y buscar datos.

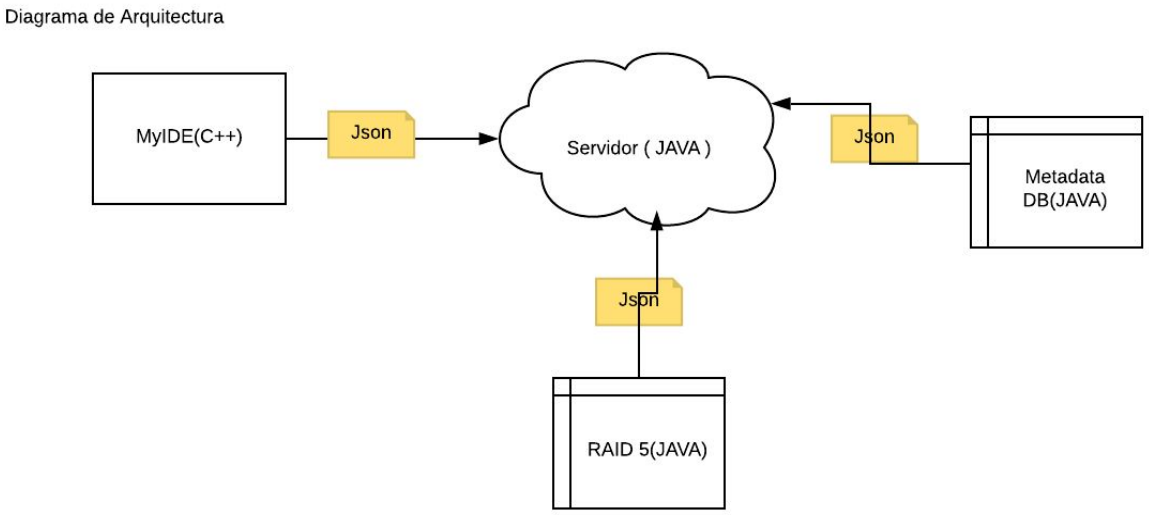
- Yo como como usuario quiero que el sistema pueda estar conformado por un conjunto de álbumes

- Yo como usuario quiero que cada galería creada esté conformada por un conjunto de imágenes.

Investigación

Con una breve investigación de las posibles soluciones para este problema se puede trabajar dos arquitecturas:

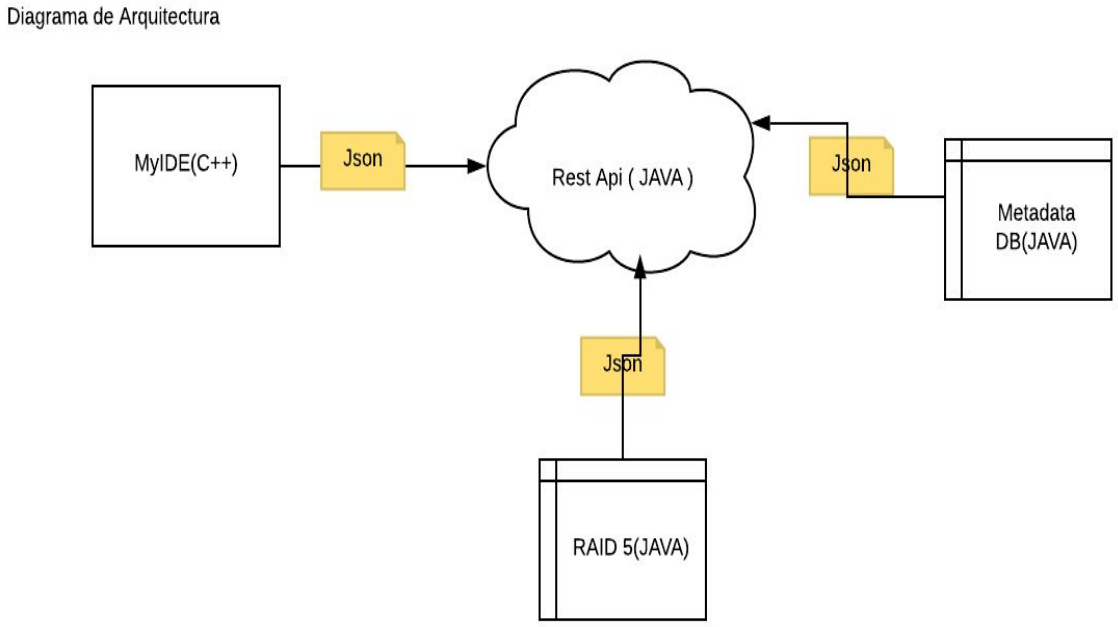
imagen 1. Diagrama de arquitectura con comunicación por medio de Sockets



En el primer caso(imagen 1) la comunicación se realizaria por Sockets, el cual se define como dos programas, quizás en diferentes dispositivos pueden intercambiar comunicación por medio de la tarjeta de red, donde este flujo es confiable y ordenada.

Esta solución es muy factible por el conocimiento en este tipo de comunicación, sabiendo que en general se utiliza un protocolo generalizado,TCP el cual no importa el lenguaje que se quieran comunicar si se utiliza esta arquitectura de comunicación, tomando en cuenta también que la red de comunicación no es necesaria que tenga la conexión a internet lo cual nos favorece, pero también tener en cuenta que el tráfico de datos no es el más veloz lo cual podría disminuir la fluidez del programa.

Imagen 2.Diagrama de arquitectura con comunicación por medio de Rest Api.



En el segundo caso(imagen 2) la comunicación se realizaria por una Rest Api , el cual se define como una API es que es la abreviatura de *Application Programming Interface*, o *Interfaz de Programación de Aplicaciones,* también se puede decir que es un contenedor de información para el intercambio entre clientes.

Esta solución no es tan factible por el conocimiento en este tipo de comunicación, sabiendo que se necesita conexión a red de tráfico de datos así como su diferencia en implementación por lenguaje así como sus peticiones, lo cual es perjudicial para nosotros por el tiempo de entrega y demás tareas que demanda tiempo, sus ventajas es la rapidez de transmisión de datos entre dispositivos los cuales eso sí depende de la velocidad de red.

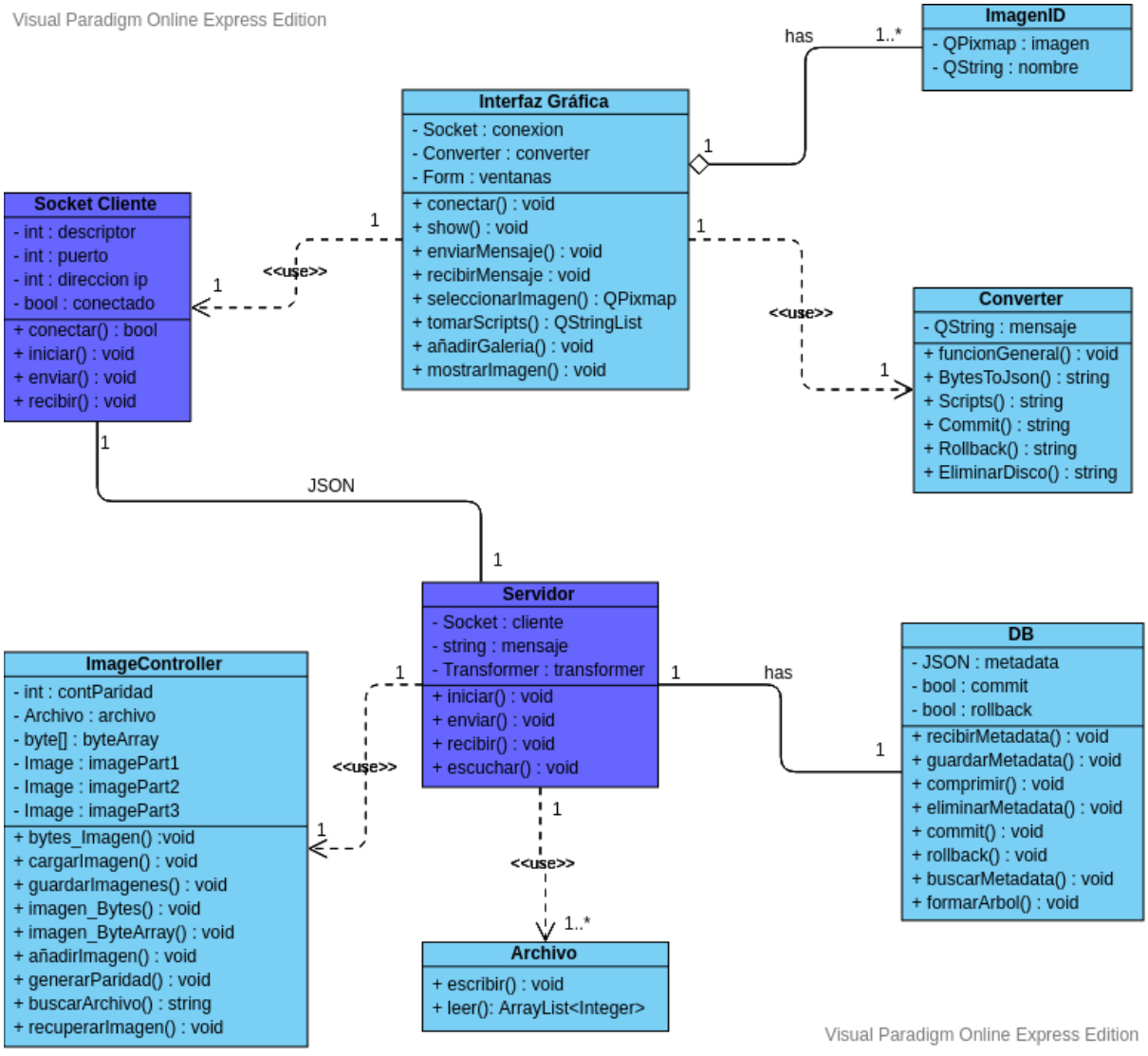
Dado a una reunión como grupo de trabajo se llego a la conclusión de la primera arquitectura(imagen 1) por diferentes motivos que se refieren a tiempo, eficiencia con respecto a conocimiento, red de conexión así como otros aspectos no tan relevantes para la decisión como grupo.

Por consiguiente los diagramas de clase y componentes quedan del siguiente modo, imagen 3 y imagen 4, en los cuales se presenta mejor la solución de la mejor manera según nuestro criterio.

Con respecto al servidor se implementará en el lenguaje de programación Java por motivo al manejo de datos con archivos, así también el uso de bases de datos no relacionadas con mayor eficiencia en conocimiento, la investigación de información para la resolución del problema se encuentra más en dicho lenguaje y el gusto por programar en ese lenguaje.

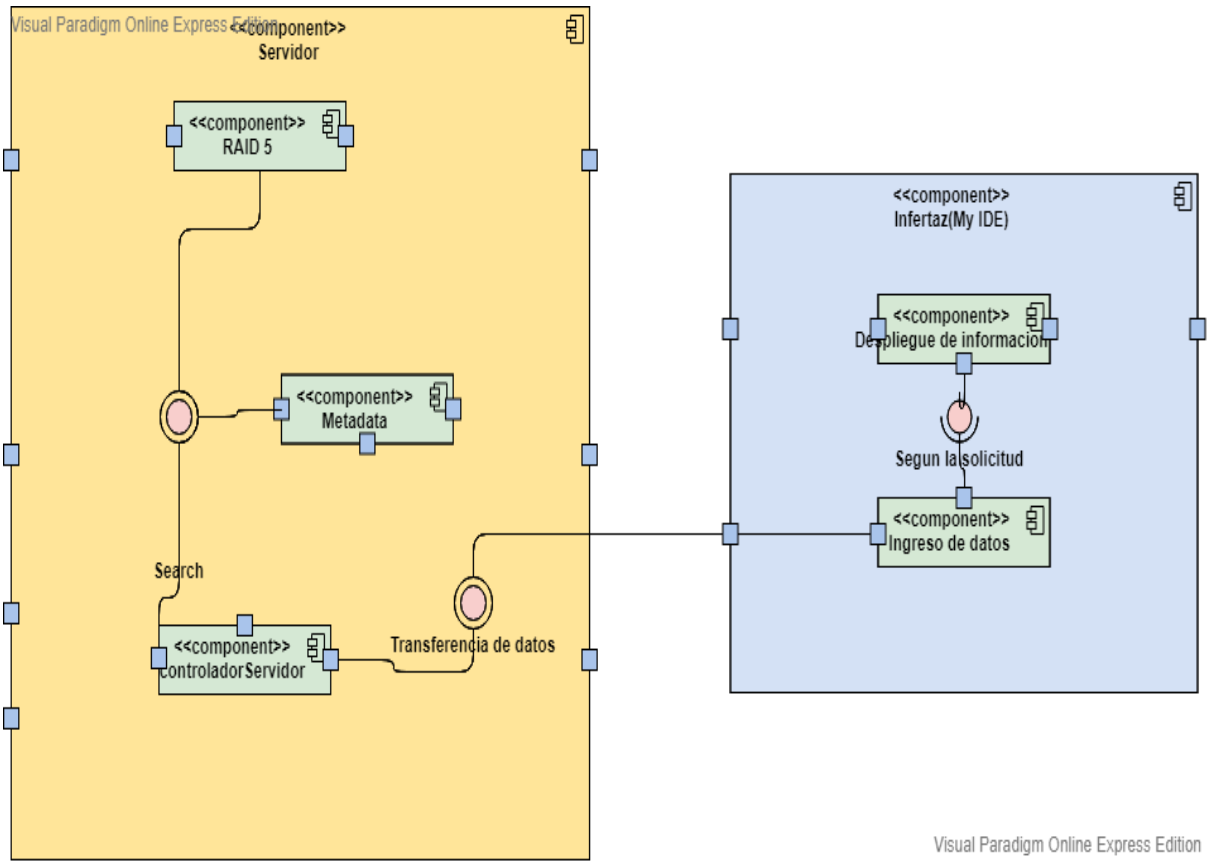
Imagen 3: Diagrama de clases

09



Siendo este el diagrama de clases(imagen 3) dando un resumen de la solución que decidimos utilizar y el diagrama de componentes(imagen 4) se puede ver mejor la agrupación de los módulos de ejecución del proyecto final.

Imagen 4:Diagrama de componentes



Evaluación grupal

Saymon		
Auto-evaluación		
Siento que para este proyecto trabajé bastante bien, cumplí con las tareas que se me asignaron, siempre me reuní con los compañeros cuando era necesario. Al menos en la parte que me tocó a mi (RAID5), le puse mucho empeño y aunque fue difícil al principio, al final se pudo lograr hacer. Igual siento que todo fue gracias a que el equipo funcionó muy bien, donde hubo mucha comunicación y apoyo mutuo.		
Co-evaluación por los demás compañeros		
Oscar	Kevin	Kennet
Logró completar de manera satisfactoria con la que considero la parte más complicada del proyecto (el manejo de imágenes). Concluyó todas sus tareas y ayudó a los otros miembros del equipo con el resto de requerimientos.	Desde el inicio puso mucho empeño en su trabajo y logró avanzar en su parte con rapidez y eficiencia. Además, siempre estuvo dispuesto a ayudar a los demás si habían problemas.	Siempre muy centrado en sus trabajos siendo demasiado bueno en sus trabajos y la efectividad de su código, muy buen compañero y siempre dispuesto a ayudar y colaborar en todos los aspectos, muy buen amigo.

Kevin		
Auto-evaluación		
Elegí la parte de interfaz gráfica ya que me gustó la idea de acceder a otros sistemas a través de widgets. Siento que hice un buen trabajo, funcional y además atractivo con un entorno jurásico. Además he escuchado las opiniones de mis compañeros desarrolladores para detalles en estética y funcionamiento que me ayudaron a tener una interfaz más funcional. También me dediqué a hacer todas las pruebas que mis compañeros necesitaron.		
Co-evaluación por los demás compañeros		
Oscar	Saymon	Kennet
Completó toda la interfaz por su cuenta. Trabajó la	Siempre se comprometió a trabajar, cumplió con lo	Persona muy centrada a la hora de trabajar con

comunicación entre Java y C++. Estuvo siempre receptivo a críticas y fue un miembro importante durante el proceso de testing.	que se le asignó y se notó el esfuerzo que hizo por lograrlo	una , firma especial en sus trabajos donde se remarca más en interfaces gráficas, es muy dedicado y buen compañero amigo.

Oscar		
Auto-evaluación		
Siento que trabajé de manera satisfactoria, y que pude cumplir con las tareas que me fueron asignadas durante la repartición del trabajo. Me encargué de la BaseDeMetadata, el algoritmo de compresión de Huffman, la funcionalidad de commit y rollback, y de la comunicación entre el servidor Java y el cliente en C++.		
Co-evaluación por los demás compañeros		
Saymon	Kevin	Kennet
Fue un compañero que desde el principio fue ambicioso y fue el que se puso a trabajar en los puntos extra, lo cual está excelente. Siempre se esforzó con las tareas que se le asignaron y cumplió con ellas de muy buena manera.	Le dedicó mucho esfuerzo y logró obtener resultados muy eficientes. Siempre dispuesto a reunirse y hacer todas las pruebas con los demás. Por otra parte, siempre estuvo centrado en trabajar duro y de la mejor manera.	Es un compañero muy dedicado al trabajo donde sus prioridades se notan muy bien demarcadas y cumplio muy bien sus metas establecidas en el proyecto y muy buen amigo.

<table>
<tr><td colspan="3">Kennet</td></tr>
<tr><td colspan="3">Auto-evaluación</td></tr>
<tr><td colspan="3">Elegí la parte de comunicación por el motivo a que es donde he trabajado más, empecé con Rest api pero la cual dio muchos problemas por motivos de problemas de hardware de mi computadora lo cual hizo el cambio de comunicación, soy bastante hiperactivo a la hora de trabajar, me cuesta concentrarme en un solo objetivo y me gusta aportar ideas en general y saber lo que hacen para mantener una idea del trabajo, pero en este proyecto no fue lo suficientemente eficiente como podía ser para el equipo.</td></tr>
<tr><td colspan="3">Co-evaluación por los demás compañeros</td></tr>
<tr><td>Oscar</td><td>Kevin</td><td>Saymon</td></tr>
<tr><td>Estuvo atento al trabajo en grupo y a las reuniones de trabajo. Participó aportando ideas para solucionar problemas en el sistema.</td><td>Un poco bullicioso pero su rol de líder permitió evacuar las dudas importantes. Facilitó muchas soluciones a problemas y brindó ideas que ayudaron a tener un código más eficiente.</td><td>Como líder del grupo fue el que comunicaba las dudas que teníamos con el profesor. Si teníamos una duda, de inmediato se la hacía saber al profesor. Siempre estuvo dispuesto a reunirse a trabajar y no puso ningún “pero”.</td></tr>
</table>