Instituto Tecnológico de Costa Rica
Area Académica de Ingeniería en Computadores
Curso: CE2103 Algoritmos y Estructuras de Datos II
Profesor Isaac Ramirez

Proyecto #3

TEC Media File System

Estudiantes:

Christopher Arredondo Fallas
2016094916

Allan Navarro Brenes
200943530

Erick Obregón Fonseca
2016123157

II Semestre 2017

Indice.

1. Introducción.

En el presente documento se presenta la planificación realizada para el proyecto “TEC Media File System”, el cual puede describirse de manera sencilla como un servicio Streaming que permite a un cliente enviar videos y consultarlos cada vez que lo requiera. Cada video llega a un nodo principal ControllerNode que se encarga de dividirlo en partes más pequeñas, y distribuirlo a los nodos de almacenamiento DiskNodes, el sistema funciona como un RAID 5, por lo que en el caso de que un nodo falle, se puede recuperar la información de acuerdo con la paridad almacenada de manera distribuida en los nodos.

2. Desarrollo

2.1. RAID 5.

Es importante iniciar con una breve explicación del sistema RAID 5. Para este proyecto se utilizaron 4 nodos que simulan 4 discos de almacenamiento. La siguiente imagen describe cómo se almacena la información en estos. Cada archivo se divide en partes iguales de 512 kB a los cuales se les llamará bloques. Cada bloque se almacena en orden en cada disco y la paridad se almacena de manera distribuida, quiere decir que no siempre el mismo nodo se encarga de la paridad, esto para evitar que si un nodo se desconecta, todas las paridades se pierdan. Cada archivo puede abarcar varios stripes y en este caso se programó que dos archivos no pueden compartir un stripe para simplificar el código.

Stripe Number	Disk 1	Disk 2	Disk 3	Disk 4
Stripe 1	parity 1	1	2	3
Stripe 2	5	parity 2	6	7
Stripe 3	9	10	parity 3	11
Stripe 4	13	14	15	parity 4
Stripe 5	17	18	19	20

Figura 1. Distribucion de los bloques en RAID 5. Tomado de: [1]

Como lo menciona [1] este arreglo requiere un mínimo de 3 a un máximo de 32 discos. Si uno de los discos falla, la información no se piere, ya que se aprovecha la función XOR que sirve para calcular la paridad de la información proveniente de los demas bloques del stripe y esta función tiene la peculiaridad que su resultado sirve para reconstruir la información original con la que se creó, siendo necesario sólo se haya perdido uno de los bloques y que se tenga la paridad calculada.

Para este proyecto un bloque de video se lee en bytes y se almacena en un arreglo de chars. Para cacular la paridad, se toman tres bloques y cada char se convierte en 8 bits, se realiza la operacón XOR entre los tres bytes de los tres bloques y se almacena en un cuarto array de chars el cual se almacena como un bloque más, este bloque obviamente no es reproducible ya que no corresponde a ninguna codificación de video a diferencia del resto de bloques. La siguiente figura resume este proceso:

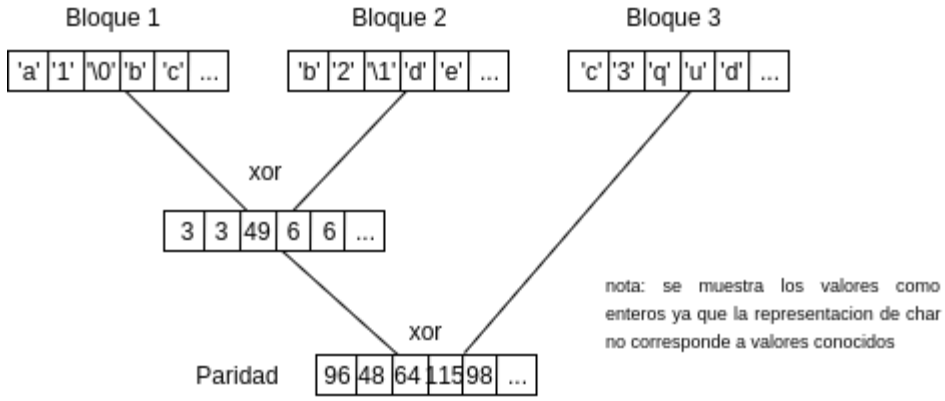

Figura 2. Calculo de paridad a partir de 3 bloques

2.2. Descripción del problema.

El proyecto consta de tres partes principales:

- **ControllerNode:** funciona como un servidor que recibe videos mediante un socket, divide el video en bloques de 512kB. A este nodo le corresponde también calcular la paridad correspondiente para los bloques y los envía mediante otro socket al siguiente componente.
- **DiskNode**: funciona también como servidor, su único cliente es el ControllerNode y se encarga de almacenar o retornar el bloque que el nodo controlador le indique. Este nodo no tiene informacion de los usuarios ni de qué videos se están almacenando.
- **Aplicación de prueba:** se encarga de buscar videos en la carpeta especificada por el usuario, enviar cada video completo al nodo controlador. Además consulta los videos que tiene el usuario y los puede reproducir. El reproductor permite pausar el video o detenerlo completamente.

2.3. Planificación y administración.

2.3.1. Features.

A nivel general se tienen los siguientes Features generales.

- Interfaz para visualizar y subir videos.
- Almacenamiento redundante.

- Coordinación entre interfaz y almacenamiento.

2.3.2. Historias de usuario.

Las features definen una clasificación para las historias de usuario, como se muestra en la siguiente figura.

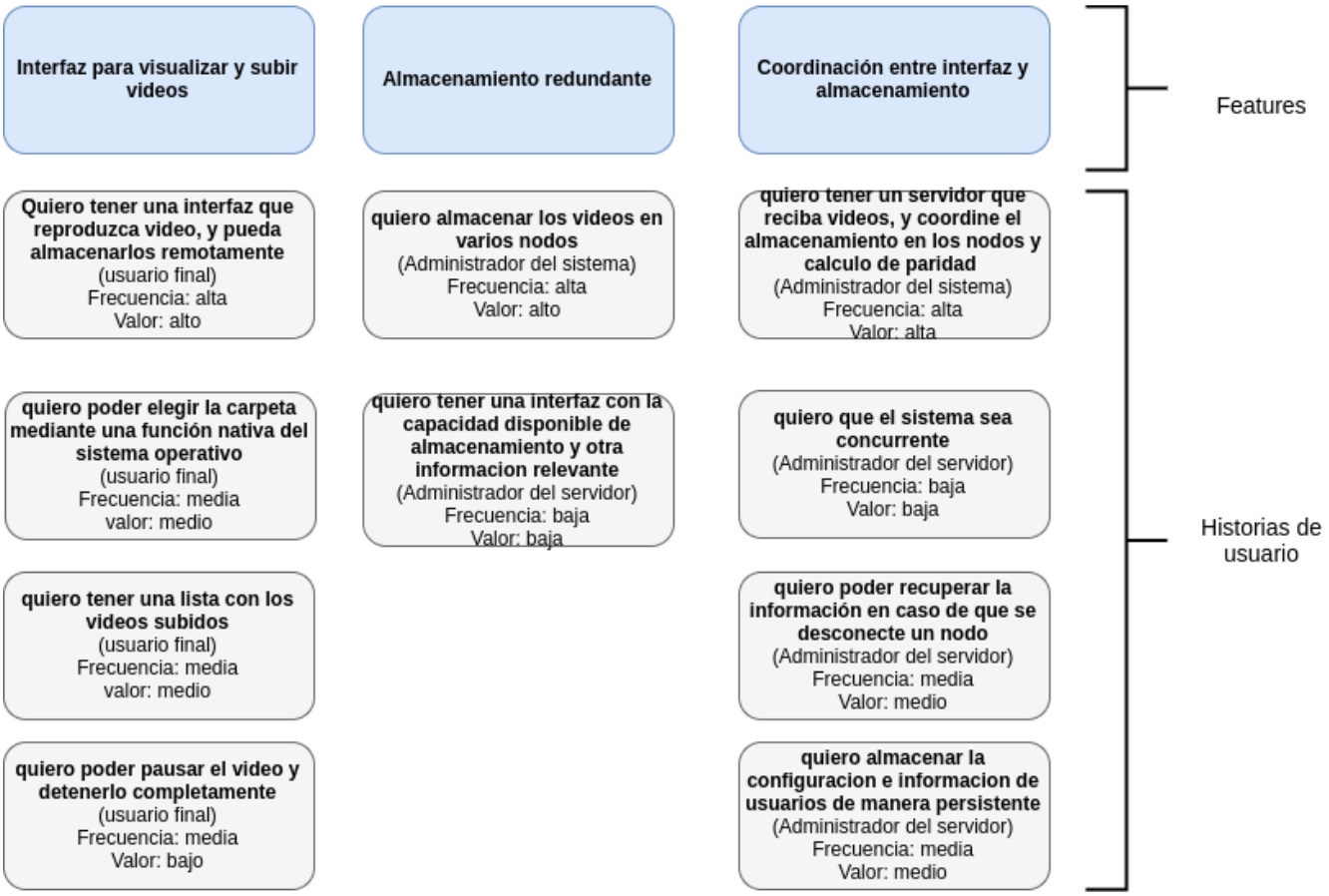

Figura 3. Historias de usuario.

2.4. Distribución según secuencia y criticidad y Minimal System Span

En la siguiente figura se realiza una distribución por secuencia y criticidad como lo menciona la referencia [2]. Este autor también menciona al minimal sistem span, el cual es la mínima cantidad de historias de usuario que permiten tener una aplicación funcional Para este proyecto, estas historias coinciden con el primer nivel de criticidad, como se muestra en la siguiente figura.

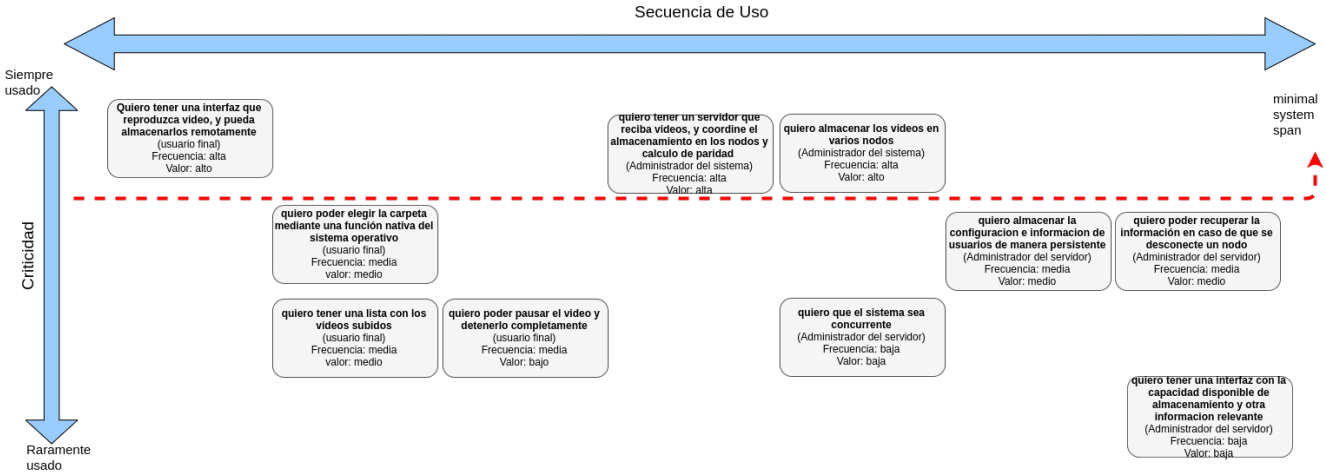

Figura 4. Distribucion según secuencia y criticidad yMinimal system span.

2.5. Plan de Iteraciones

A continuación se muestra una figura donde se proponen distintas iteraciones de acuerdo con las historias de usuario definidas, la idea es introducir paulatinamente nuevas funcionalidades a la aplicación según la importancia que se le asignó a cada historia.

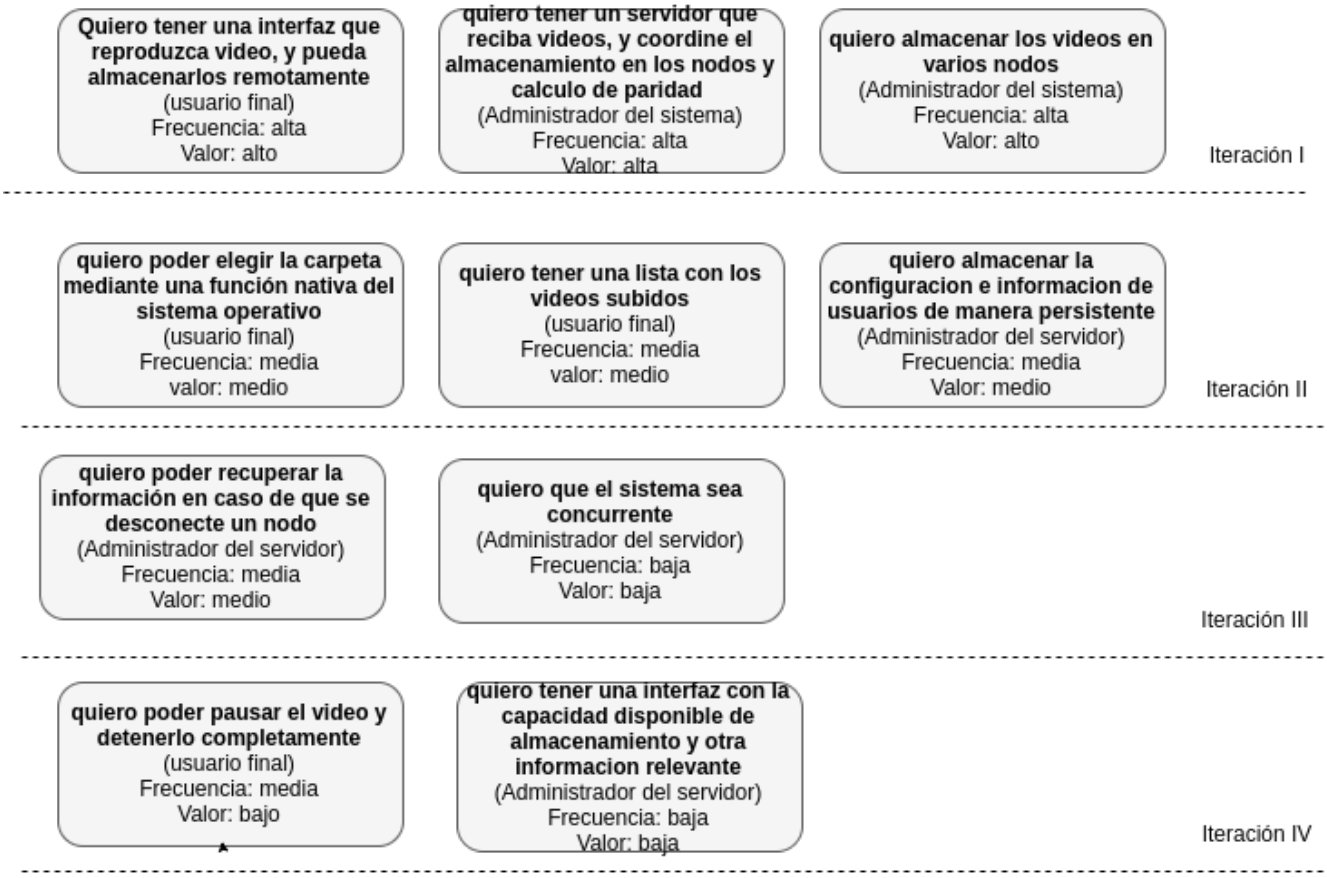

Figura 5. Plan de iteraciones.

2.6. Distribución de historias por miembro del equipo, descomposición en tareas y bitácora

En la siguiente tabla se propone la distribuión de user stories por miembro del equipo y simultáneamente se descomponen las historias en tareas. Además se indica el tiempo inverticdo y las fechas de inicio y finalización de las actividades a manera de bitácora del proyecto.

Tabla 1. Bitácora del proyecto.

Tipo	Descripción	Encargado	Iteración	Duración (horas)	Inicio	Fin
Historia	quiero tener un servidor que reciba videos, y coordine el almacenamiento en los nodos y calculo de paridad	A.N	1		06/11/17	13/11/2017
Tarea	Definir los sockets			3		
Tarea	Programar funcion de paridad			1		
Tarea	Definir programa para division de videos e integrarlo a la logica			0.5		
Tarea	Definir protocolo de comunicación entre programas			0.5		
Tarea	Realizar pruebas			1		
Historia	quiero almacenar la configuracion e informacion de usuarios de manera persistente	A.N	2		11/11/17	12/11/17
Tarea	Buscar librerias (boost)			0.5		
Tarea	Programar almacenamiento persistente de la configuracion del servidor y los usuarios y videos			2		
Historia	quiero poder recuperar la información en caso de que se desconecte un nodo	A.N	3		13/11/2017	13/11/2017
Tarea	Incluir la logica en el código			2		
Tarea	Pruebas de verificación			1		
Historia	quiero que el sistema sea concurrente	A.N	3		13/11/2017	13/11/2017
Tarea	Sincronizar servidores			1		
Historia	Quiero tener una interfaz que reproduzca video, y pueda almacenarlos remotamente	C.A	1		13/11/17	17/11/17
Tarea	Definir clases			3		
Tarea	Modelar la interfaz			4		
Tarea	Darle funciones a los botones			1		
Historia	Quiero poder elegir la carpeta mediante una funcion nativa del sistema operativo	C.A	2		18/11/17	20/11/17
Tarea	Investigo sobre librerias			1		
Tarea	Implemento librerias			2		
Tarea	Modifico para filtrar mp4			1		
Historia	Quiero poder pausar el video, ademas de poder detenerlo completamente	C.A	3		21/11/17	25/11/17
Tarea	Investigo sobre librerias			1		
Tarea	Implementacion de librerias para video			3		
Tarea	Agrego botones para pausar, iniciar y parar.			2		

Historia	Quiero almacenar video en varios nodos	E.O.F	1		11/18/17	11/26/17
Tarea	Leer XML con la configuraciones del servidor			2		
Tarea	Crear Servidor por sockets y configurarlo			6		
Tarea	Recibir y guardar videos del ControllerNode			4		
Tarea	Regresar el video solicitado por el ControllerNode			3		
Tarea	Almacenar los bloques de memoria y definir la cantidad máxima del DiskNode			1		
Historia	Quiero tener una GUI con la capacidad disponible de almacenamiento y otra información relevante	E.O.F	2		11/26/17	11/26/17
Tarea	Mostrar la información de la configuración del DiskNode			1		
Tarea	Mostrar la capacidad actual de almacenamiento del DiskNode			1		

2.7. Diseño

Se muestran a continuación los diagramas de arquitectura, componentes y clases.

2.7.1. Diagrama de Arquitectura

La siguiente figura muestra el diagrama de arquitectura de la solución. Se tiene la aplicación de prueba la cual es la interfaz que usa el usuario final, esta se comunica mediante sockets con el nodo principal el cual a su vez se comunica con los discos de almacenamiento.

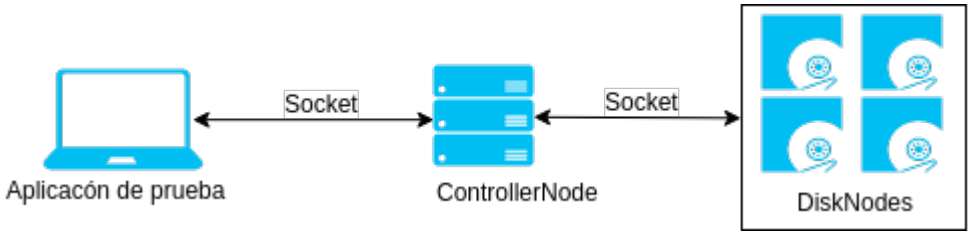

Figura 6. Diagrama de Arquitectura.

2.7.2. Diagrama de componentes

El siguiente diagrama muestra con mayor detalle los distintos paquetes que se tienen en el proyecto. Se tienen tres grandes componentes: la aplicación de prueba, el nodo controlador y los nodos disco, los cuales a su vez usan otras bibliotecas como QT, boost, y sockets.

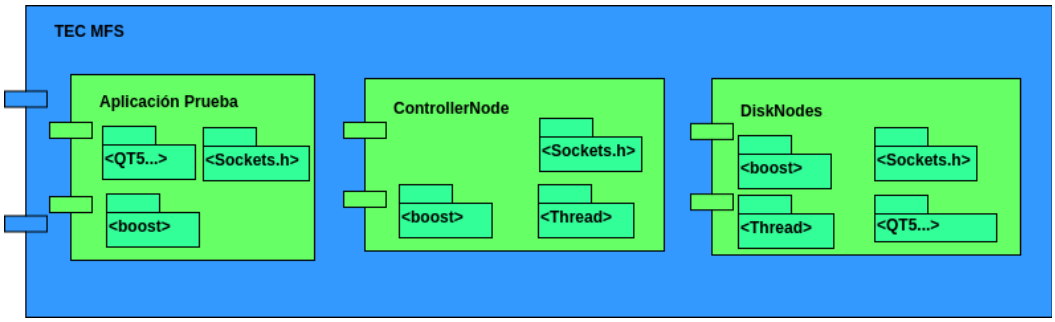

Figura 7. Diagrama de componentes

2.7.3. Diagrama de clases

A continuación se presenta el diagrama UML de clases para el proyecto.

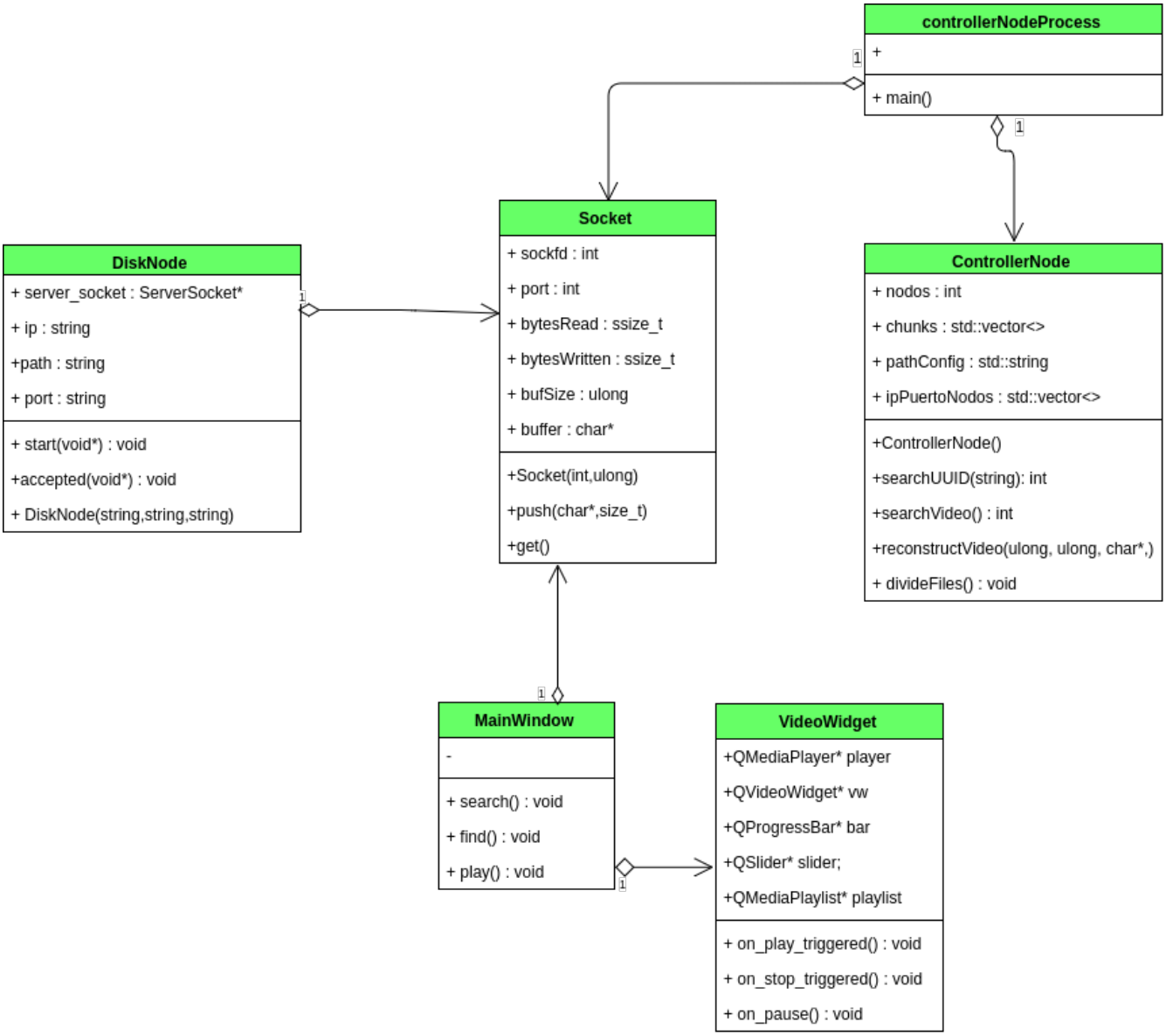

Figura 8. Diagrama de clases.

2.7.4. Diagrama de despliegue

A continuación se presenta el diagrama de la implementación de la solución en cuanto a equipos que podrían utilizarse.

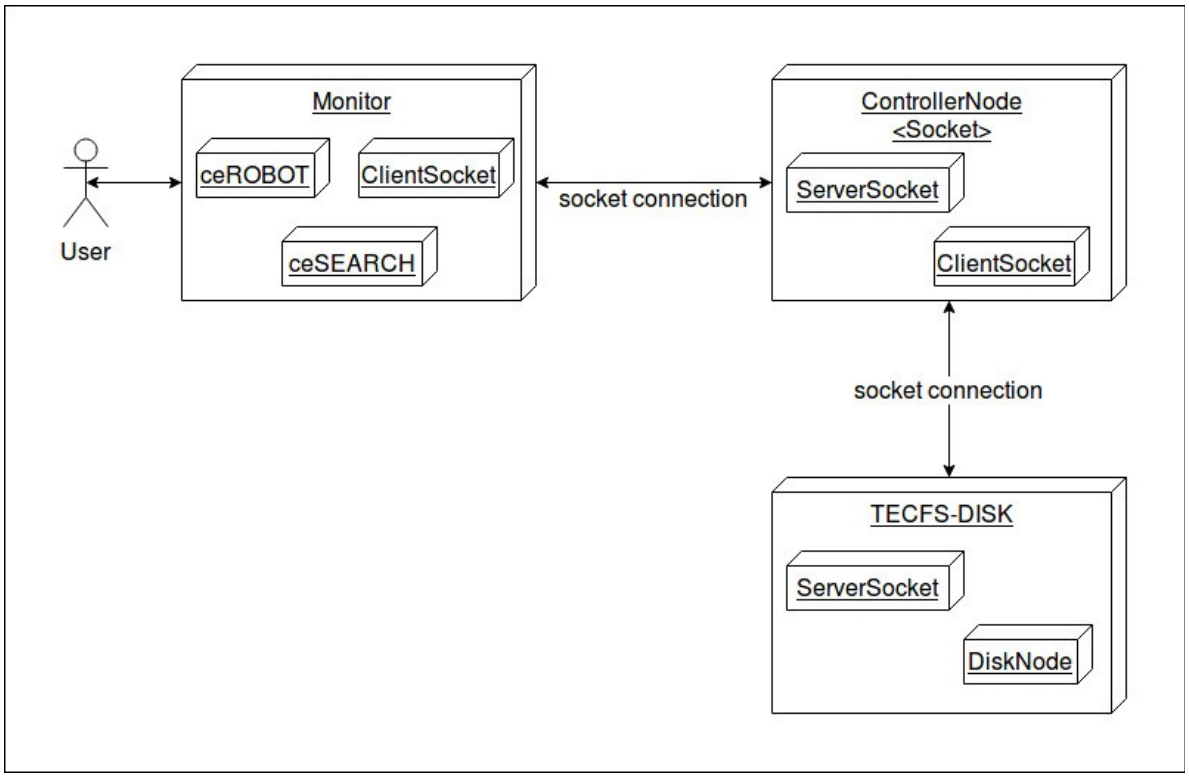

Figura 9. Diagrama de despliegue.

2.7.5. Diagrama de secuencia.

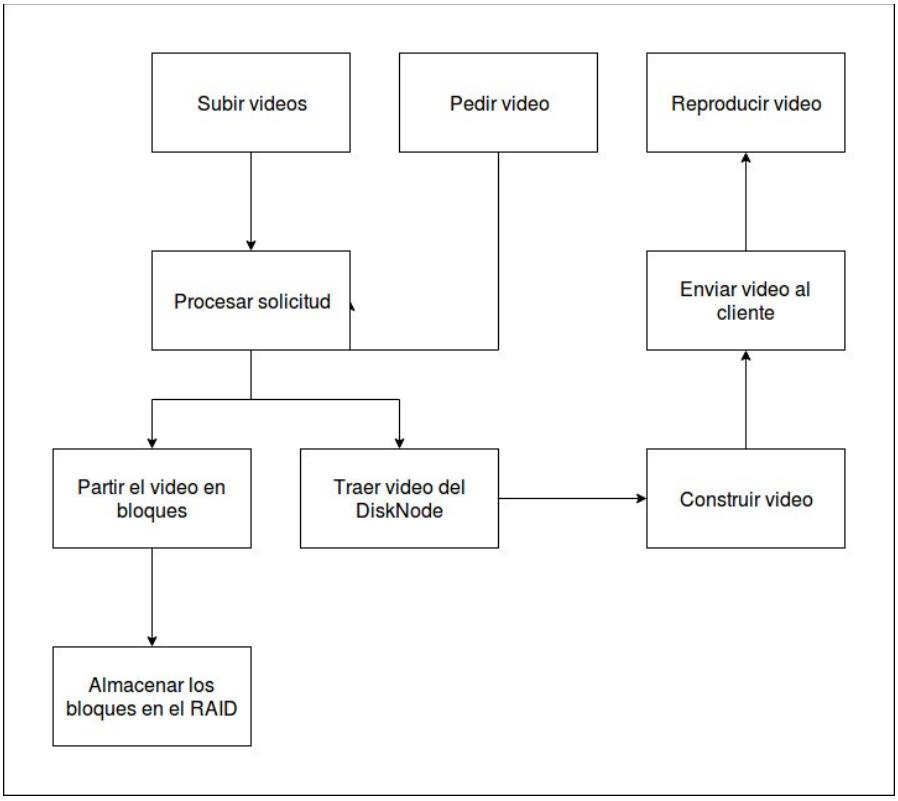

Figura 10. Diagrama de secuencia

2.8. Implementación

2.8.1. Bibliotecas utilizadas

- C++11
- Biblioteca de Sockets [3]
- boost[4]: se utiliza para manipular carpetas (crear, eliminar, contar archivos)

Otras herramientas

- CLion. [5]

2.8.2. Estructuras de Datos desarrolladas

Para este proyecto las estructuras de datos utilizadas son vectores de c++ para almacenar la información de los videos de los usuarios. No se programó ninguna otra estructura adicional.

2.8.3. Descripción detallada de los algoritmos desarrollados.

A continuación se muestran los principales algoritmos que permiten el funcionamiento del sistema.

Uno de los principales algoritmos es el que se utiliza para calcular la paridad de los bloques. Este recorre el array de bytes de los bloques y realiza la operación de xor a cada carecter de estos.

```
char* Funcs::calcularParidad(char* bytes1, char* bytes2, char* bytes3,
ulong size){
    auto paridad = new char[size]();
    for (auto i =0; i < size; i++){
        paridad[i] = (char)(std::bitset<8>(bytes1[i]) ^
std::bitset<8>(bytes2[i]) ^ std::bitset<8>(bytes3[i]) ).to_ulong();
    }
    return paridad;
}
```

Otro algoritmo importante es el de cargar un archivo a un vector de bytes, para el cual se usa la biblioteca fstream de c++

```
//abre el archivo de la computadora
auto inputFileStream= new std::ifstream(nombreVideoFullPath,
std::ios::binary|std::ios::ate);
auto tamanoVideo= (ulong) inputFileStream->tellg();
if(!inputFileStream->is_open() || inputFileStream->fail()){
    return -1;
}
//array donde se almacenaran los bytes del video
std::vector<char> resVector(tamanoVideo);
//se busca el inicio del archivo
inputFileStream->seekg(0, inputFileStream->beg);
//se almacena el archivo en el array de bytes
inputFileStream->read(&resVector[0], tamanoVideo);
inputFileStream->close();
```

2.9. Problemas encontrados

Al momento de la entrega de este documento no se encontraron problemas que puedan afectar el cumplimiento completo de los requerimientos para la entrega final del proyecto.

3. Conclusiones

- Se logró desarrollar un sistema que permite el uso de una interfaz para reproducción de videos almacenados de manera remota, además esta permite almacenar y consultar videos del usuario.
- El sistema asegura alta disponibilidad, si existe una falla en un nodo, es posible recuperar la información. Esto se logra implementando un arreglo similar al RAID 5.
- Los programas utilizan sockets y pueden comunicarse entre ellos ya sea en una misma máquina así como en varias conectadas a la misma red o remotamente mediante internet.

4. Bibliografía

[1] Microsoft. (s.f) RAID 5 Volumes, Consultado en: https://technet.microsoft.com/en-us/library/cc938485.aspx

[2] Jeff Patton. (2005). It's all in How you slice it.. Consultado en: http://jpattonassociates.com/wp-content/uploads/2015/01/how_you_slice_it.pdf
[3] R. Tougher, "Linux Socket Programming In C++, " Consultado en: http://www.tldp.org/LDP/LG/issue74/tougher.html
[4] Boost (2017) Boost c++ Libraries. Consultado en: http://www.boost.org/
[5] JetBrains, (2017) . CLion, Consultado en: https://www.jetbrains.com/

Anexos

Anexo 1. Vistas de la aplicación.

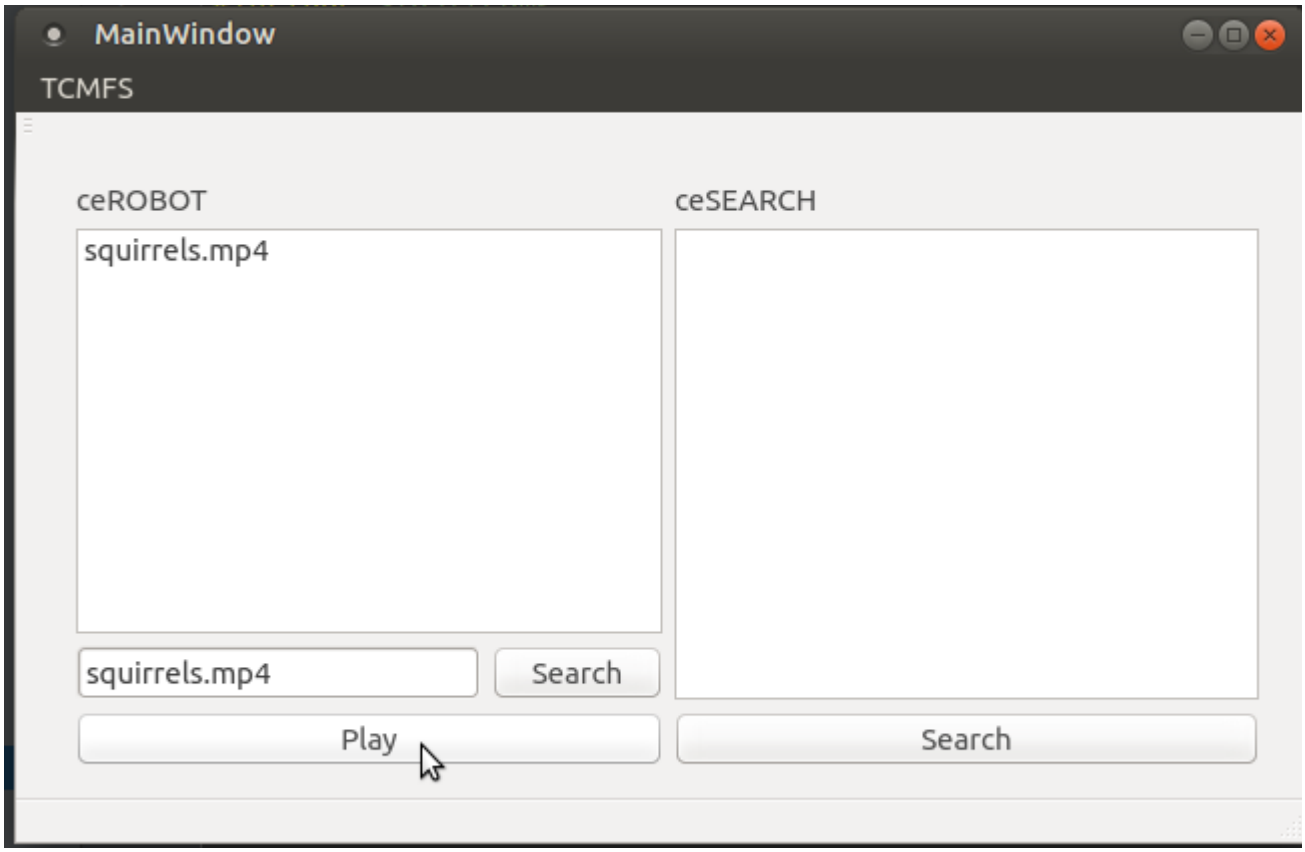

Figura 11. Interfaz gráfica.

Figura 12. Reproductor