UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Almacenamiento y visualización de datos del radio-observador del IGEF.

TESIS

Que para obtener el título de

Ingeniera en Computación

P R E S E N T A

Yessica Gisela Arredondo Guzmán

DIRECTOR DE TESIS

Dr. Alejandro Lara Sánchez

Ciudad Universitaria, Cd. Mx., 2016

Dedicatoria

A mi madre, Patricia Guzmán Guzmán.
Por su amor infinito, su apoyo incondicional en cada etapa de mi vida, cada cosa que hago es gracias a la dedicación que ha puesto desde siempre en mi...

A mi hermano, Mario Alberto Arredondo Guzmán.
Por su reconfortante compañía, sus consejos, sus bromas y sus juegos...

Agradecimientos

A mi director de tesis, **Dr. Alejandro Lara**, por su paciencia y confianza que deposito en mi para la realización de este trabajo.

Al **M.I. Mario Alberto Díaz Cruz**, por su colaboración y constante apoyo durante el desarrollo de este trabajo.

Al **Sr Filiberto Matias**, por la disposición que brindó durante mi estancia en el laboratorio del IGEF.

Le agradezco infinitamente a la **Universidad Nacional Autónoma de México** por el honor que me ha brindado de pertenecer a ella.

Por mi raza hablara el espíritu.

Este trabajo fue posible en parte gracias al apoyo de CONACYT 179588 y DGAPA IN111716-3.

Índice general

Capítulo 1

Introducción

1.1. Resumen

El Radio Observatorio Solar del Instituto de Geofísica de la UNAM, cuenta con cuatro instrumentos, cada uno almacena sus datos en un formato propio y son guardados en su correspondiente servidor.

Para homogeneizar los datos, se requiere elaborar un sistema que unifique el formato de estos archivos al formato "Flexible Image Transport System" mejor conocido por sus siglas en ingles como FITS y se administren por medio de una base de datos para ser consultados y descargados en un portal de internet.

Este portal debe contar con la visualización inmediata de los archivos que se están generando en tiempo real y debe contener además una sección donde se puedan consultar mediante un calendario los datos que se han ido guardando a través del tiempo, con la opción de descarga de los archivos FITS.

FITS es un formato de archivo digital que es utilizado por los astrónomos, se usa principalmente como un método para intercambiar datos de mapa de bits entre diferentes plataformas de hardware y aplicaciones de software.

Básicamente, un archivo FITS está estructurado con una cabecera, normalmente seguida por un flujo de datos, este flujo de datos puede ser de dos tipos: imágenes o tablas de datos.

Este trabajo consiste en la implementación de un sistema que todos los días guarda los archivos de dos de los instrumentos del observatorio: RIS y CALLISTO.

Para ello se requiere configurar un servidor nombrado Andrómeda que tendrá tanto el desempeño de servidor web como de servidor de base de datos.

En el primer capítulo se presenta el análisis del escenario inicial como son: la configuración de la red de datos del laboratorio, el formato de los archivos que se manipulan y las necesidades que tenía que satisfacer el sistema que se desarrollará

El segundo capítulo trata sobre la recepción y unificación de los archivos FITS: RIS cuenta con archivos xdat que deben ser convertidos a archivos FITS y colocados en una carpeta de recepción en Andrómeda, por su parte CALLISTO cuenta con archivos FITS que deben ser colocados al igual que los del RIS en el servidor Andrómeda.

En el tercer capítulo se presenta el análisis de los datos que definen los atributos de cada archivo para el diseño e implementación de la base de datos que nombraremos en adelante como base de datos Geofísica.

El algoritmo para la carga de datos tanto históricos como diarios de los dos telescopios se detalla en el cuarto capítulo.

Posteriormente se presenta, en el quinto capítulo, el algoritmo para la visualización de los datos en el portal web http://cintli.geofisica.unam.mx del Radio Observatorio Solar: para el telescopio RIS se cuenta con un archivo FITS que contiene una tabla de datos, mismos que se mostrarán en una gráfica que se ejecutará del lado del cliente utilizando una biblioteca de java script llamada Jcharts. CALLISTO tiene archivos FITS que contienen una imagen, esta imagen se crea con Python del lado del servidor al momento de la petición de la página.

El sexto capítulo aborda la configuración del entorno de trabajo del servidor Andrómeda para que el sistema trabaje correctamente.

De ahora en adelante llamaremos Sistema Andrómeda, a los archivos de configuración, a las dos bases de datos y al conjunto de programas que se desarrollaron en diferentes lenguajes de programación.

1.2. Objetivo

El objetivo de este trabajo es:

a) Automatizar la transferencia de los archivos que generan RIS y CALLISTO todos los días al servidor Andrómeda.

b) En Andrómeda serán procesados para unificar su formato a archivos FITS.

c) Los archivos FITS serán guardados en Andrómeda y se administrarán por medio de una base de datos.

d) Los archivos FITS tanto de RIS como de CALLISTO se podrán consultar en tiempo real y de manera histórica en un portal web diseñado para el Laboratorio del Radio Observatorio Solar del Instituto de Geofísica de la UNAM.

1.3. Aportación

Inicialmente, en el laboratorio sólo se guardan los datos adquiridos por los telescopios en el servidor que les corresponde sin la oportunidad de que puedan ser consultados y manipulados para su estudio por personas ajenas al laboratorio.

En el caso de RIS, sus archivos originales están en un formato que no es comprensible para la mayoría de las personas ya que es un formato único que fue creado por un estudiante.

Este trabajo unificará los formatos a archivos FITS que es un formato estándar comprensible para los científicos dedicados a estas áreas.

El laboratorio no cuenta con un portal web por lo que se diseñará uno donde se podrán consultar los archivos en su representación gráfica y se podrán descargar los archivos FITS por cualquier persona interesada en estudiar estos datos.

Con el presente trabajo el Radio Observatorio del Instituto de Geofísica de la UNAM ganará visibilidad y así podrá compartir sus datos y observaciones con la comunidad interesada.

Capítulo 2

Análisis del escenario

El laboratorio del Radio Observatorio Solar del Instituto de Geofísica de la UNAM cuenta con cuatro instrumentos que están monitoreando la actividad solar permanentemente, el objetivo del Sistema Andrómeda consiste en centralizar los datos arrojados cada día de dos de ellos: RIS y CALLISTO, con el fin de unificar sus formatos y tener un fácil acceso a su contenido.

Los telescopios son los siguientes:

1. Radio Interferómetro Solar (RIS).
2. Espectrómetro CALLISTO.
3. Estación Radio JOVE.
4. Estación LAVNET-México.

RIS y CALLISTO diariamente obtienen datos y cada uno tiene un formato de salida diferente, el objetivo principal del Sistema Andrómeda es unificar el formato de datos obtenidos por cada uno de ellos, administrarlos con una base de datos y mostrarlos al público en un portal web tanto de manera histórica, como en tiempo real.

La configuración inicial de la red del laboratorio que se muestra en la figura 2.1, no tiene destinado ningún servidor que desempeñe el trabajo de centralizar los datos de los instrumentos del laboratorio, tampoco para que desempeñe el trabajo de servidor web.

Para llevar a cabo el presente trabajo fue necesario realizar la conexión de un servidor que fue nombrado Andrómeda al cual se le instaló el sistema operativo

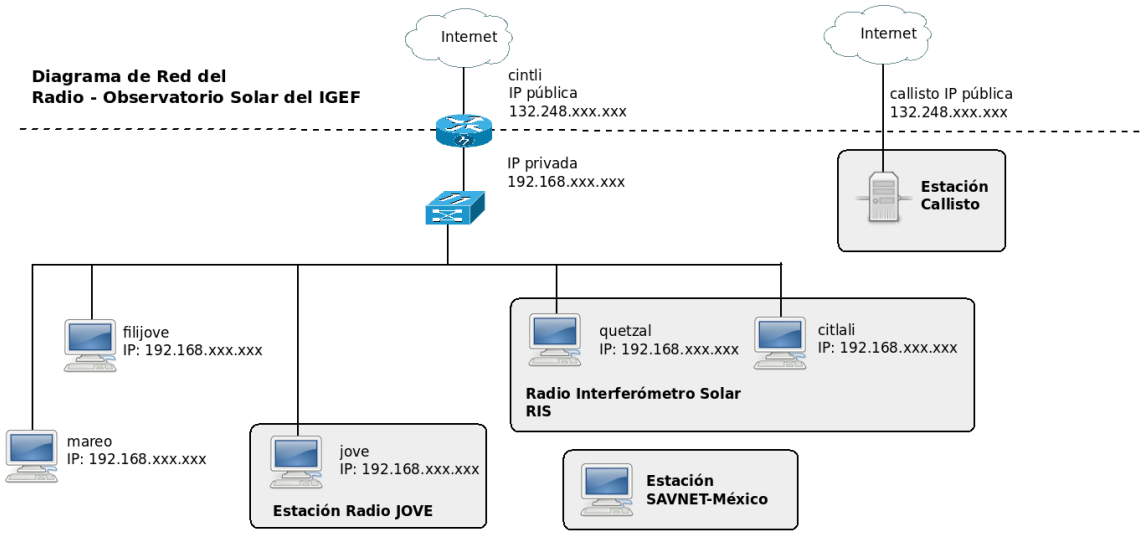


Figura 2.1: Diagrama original de Red del laboratorio del Radio Observatorio Solar del Instituto de Geofísica de la UNAM

Debian 6, ya que las computadoras del laboratorio trabajan con este sistema operativo.

Andrómeda desempeñará el papel de:

1. Servidor web.
2. Servidor de Base de datos.

Especificaciones técnicas del servidor Andrómeda:

- Procesador: 64-bit Intel® Xeon® Processor 3.60 GHz, 1M Cache.
- Memoria RAM: 3.466324 Gb.
- Disco duro: 8 Gb.
- Disco externo: 160 Gb *.

La configuración de red final se muestra en la figura 2.2, en el capítulo 6 se desarrollará la configuración de los servicios necesarios para su funcionamiento.

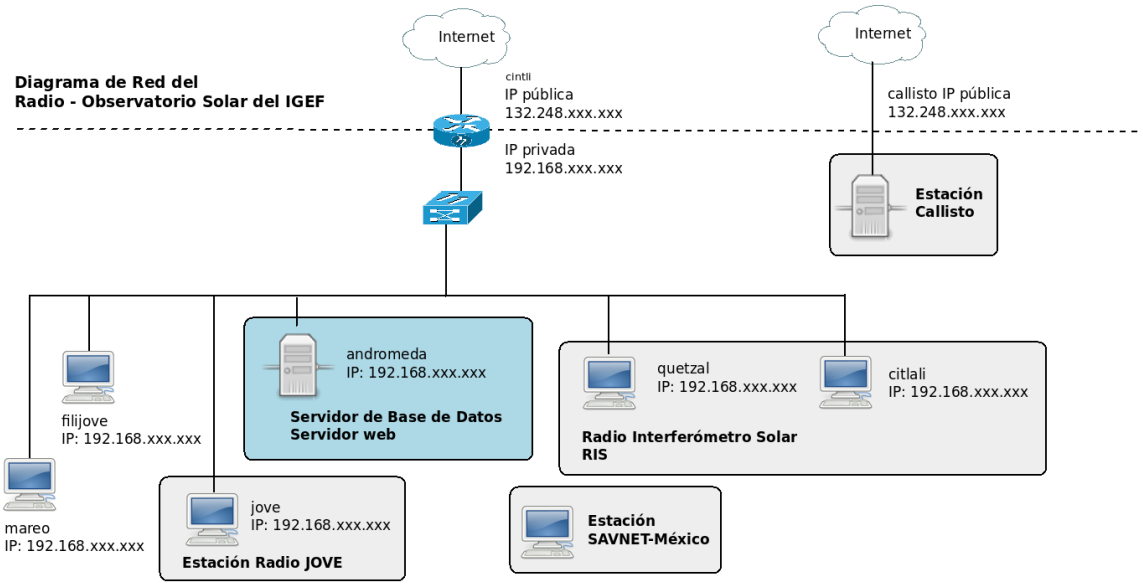


Figura 2.2: Diagrama final de Red del laboratorio del Radio Observatorio Solar del Instituto de Geofísica de la UNAM

2.1. RIS

El Radio Interferómetro Solar de base pequeña del Instituto de Geofísica (RIS) detecta la emisión solar en microondas durante el día, por un periodo de 8 horas, estas emisiones regularmente se mantienen constantes, a menos que exista una ráfaga o fulguración, es decir una erupción en la corona solar, esto indica que hay actividad en la atmósfera del sol. Las ráfagas pueden ser seguidas de eyecciones de masa coronal, y así podemos determinar el estado del clima espacial. El RIS consta de dos antenas parabólicas de 1m de diámetro, cada una montada en un eje polar común que determina la dirección de la base del interferómetro. Opera en una frecuencia central de 7.7GHz con un ancho de banda de 500MHz. La separación de las antenas es de 5.24m que corresponde a 131 longitudes de onda. El eje del aparato está orientado a lo largo del eje polar y un motor síncrono lo gira de este a oeste, haciendo girar las antenas, siguiendo el movimiento aparente del Sol. Cada antena tiene además un movimiento independiente en declinación, impulsado por un sistema adaptado de motores (para más información consultar las tesis[4] : Desarrollo de sofware para la captura, despliegue, almacenamiento, procesamiento y publicación de datos en

tiempo real obtenidos del radio interferometro solar ris).

El RIS se encuentra en la azotea del edificio principal del Instituto de Geofísica (IGEF) en Ciudad Universitaria, cuenta con una tarjeta de adquisición de datos que está conectada al servidor Quetzal, estos datos son guardados al finalizar su periodo diario de observación en el servidor Citlali, con un formato de salida `xdat` desarrollado específicamente para esta tarjeta.

La configuración del RIS se puede apreciar en la figura 2.3.

Figura 2.3: Configuración del RIS

2.2. CALLISTO

El IGEF de la UNAM recibió el espectrómetro solar CALLISTO, su nombre es un acrónimo de Enhacement-Compact Astronomical Low-Frecuency, Low-Cost Instrument for Spectroscopy in Transportable Observatories por parte del Instituto de Astronomía de Zurich, Suiza. El espectrómetro es adecuado para la observación de los estallidos solares que ocurren en la longitud de onda métrica y decimétrica.

A la frecuencia que trabaja CALLISTO podemos observar la emisión de la Corona Solar y como tenemos resolución espectral, es decir, tenemos observaciones a diferente frecuencia. podemos ver casi al mismo tiempo varias alturas en la corona. También podemos observar agentes perturbadores moviéndose por la corona, llamados estallidos de radio tipo II y tipo III.

El espectrómetro solar CALLISTO situado en la azotea del edificio principal del IGEF en Ciudad Universitaria, cuenta con una tarjeta de adquisición de datos que

se encuentra conectada al servidor CALLISTO. En dicha maquina se encuentran programas que se encargan de enviar a Suiza y guardar los archivos FITS que se generan.

La información de CALLISTO es almacenada en archivos FITS cada 15 minutos de las 13:00 hasta las 17:45 horas.

La configuración de CALLISTO se puede apreciar en la figura 2.4.

Figura 2.4: Configuración de CALLISTO

Capítulo 3

Recepción de datos

Para la recepción de archivos hay que tener en cuenta que todos los días se deben de realizar las copias al servidor Andrómeda, de los archivos generados por cada instrumento. Como se mencionó CALLISTO termina su trabajo a las 17:45 horas por lo que es conveniente realizar la recolección de datos en la noche. Para el caso de RIS en las mañanas se transmitirán los archivos `xdat` del día anterior porque una vez que RIS termina su trabajo el servidor se apaga automáticamente perdiendo comunicación con el servidor Andrómeda. Para la transferencia de los archivos se utiliza el protocolo `ssh` y el proceso se automatizó con un script shell.

`Ssh` tiene una arquitectura cliente/servidor y cada conexión la realiza de manera segura encriptando la sesión de conexión, es decir si alguien por medio de un ataque llegara a obtener la contraseña del servidor y obtuviera los datos, estos estarían encriptados y no podrían ser leídos.

Este protocolo esta implementado en un conjunto de aplicaciones de código abierto llamado OpenSSH, una de ellas es `scp` que realiza copias seguras de archivos: los archivos viajan encriptados en su paso por la red.

Para el sistema Andrómeda se consideró más apropiado implementar la autenticación por medio de llaves en lugar de contraseñas haciendo de esta forma más segura la conexión, ya que las contraseñas de los servidores no estarían escritas en los scripts shell que se encargarán de automatizar el proceso diariamente.

La autenticación por medio de llaves se explica en la siguiente figura (3.1):

En el servidor Andrómeda se crean y se guardan un par de llaves, una pública

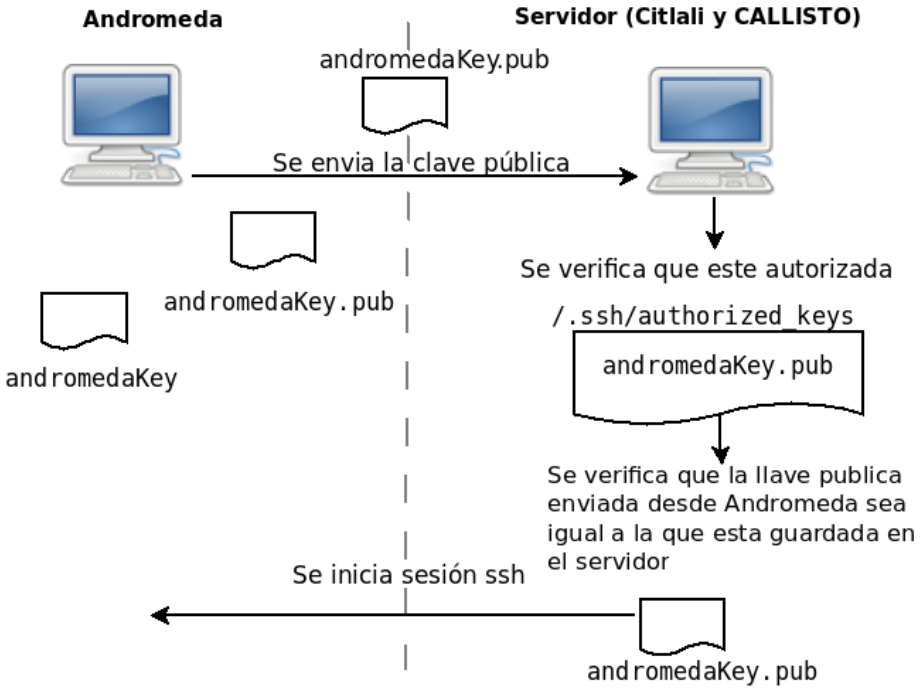


Figura 3.1: Diagrama del intercambio de llaves publicas para la autenticación SSH

`andromedaKey.pub` y una privada `andromedaKey` posteriormente se guarda una copia de la llave pública en el servidor.

Cada vez que hay una conexión Andrómeda envía `andromedaKey.pub` al servidor que se quiera autenticar. El servidor verifica que esté autorizada en un archivo de configuración donde se guardan las llaves autorizadas, después se compara la llave enviada con la guardada en el servidor. Si las dos verificaciones son ciertas, comienza la conexión `ssh`.

Cuando se ejecuta el comando `scp` para la copia de un archivo, en el lado del servidor el archivo es encriptado con la llave pública, una vez que es copiado a Andrómeda el archivo es descifrado con la llave privada.

Para automatizar la recepción de archivos se utilizaron scripts shell. El shell de GNU/Linux es un programa que es utilizado para la interacción entre el usuario y el kernel del sistema operativo, los scripts son archivos ejecutables que contienen una serie de sentencias y comandos para realizar una tarea en específico, en este caso la copia diaria de los archivos deseados.

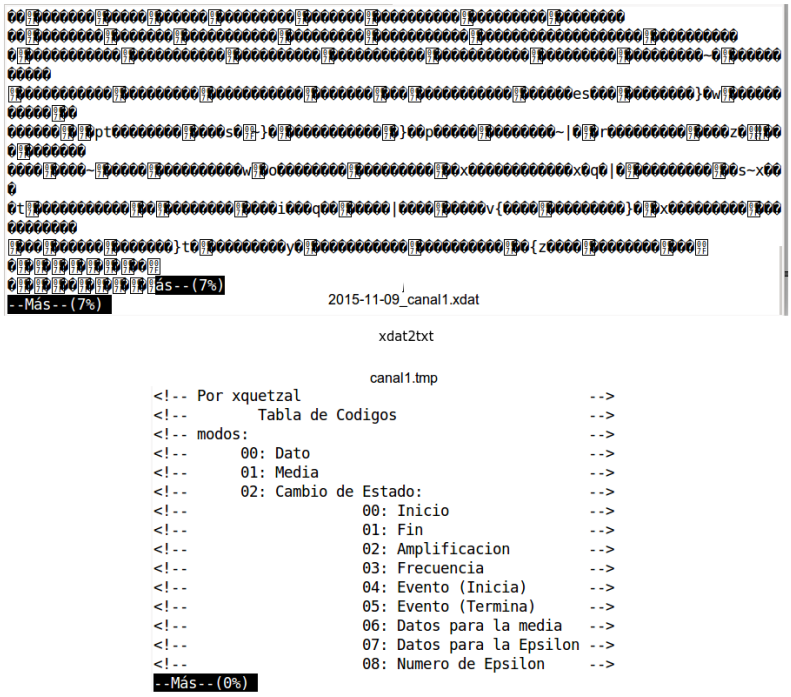


Figura 3.2: Transformación de los datos del RIS

3.1. Análisis del formato de salida de RIS

El radiotelescopio RIS, adquiere datos solares que son guardados cada día con un formato de salida llamado `xdat` y son almacenados en el servidor Quetzal.

Los archivos `xdat`, son archivos creados a partir de un software desarrollado por alumnos del laboratorio. Se crea un archivo para cada canal, es decir arroja cuatro archivos terminando su periodo diario. Los archivos `xdat` están codificados, el laboratorio solar cuenta con un programa que decodifica estos archivos llamado `xdat2txt`.

El archivo decodificado contiene una cabecera con datos informativos de su muestreo como son hora de inicio, hora en que finaliza, día que se realizó, seguida de un flujo de datos con dos columnas que representan el tiempo y la intensidad para el canal 1, la polarización para el canal 2 y la posición de la fuente emisora canal 3 y canal 4. En la figura 3.2, se muestra un bloque de caracteres codificados, una vez que pasan por la transformación de datos por medio del programa `xdat2txt` se puede ver que ya tienen una salida comprensible para el usuario.

Una de las necesidades resueltas en este trabajo es la implementación de un sistema que tome estos datos arrojados en formato `xdat` y convertirlos al formato de archivos FITS, para ser administrados por medio de una base de datos y que puedan ser visualizados vía web, donde el usuario podrá elegir una fecha y el sistema mostrará la gráfica con los datos de este telescopio del día elegido, con la opción de descargar los archivos FITS. También se podrán consultar los últimos datos registrados por el telescopio en la página principal.

3.2. Algoritmo para la recepción de los archivos del servidor RIS

La recepción de los archivos del RIS está a cargo del programa `recepcionRis.sh` que ejecuta un algoritmo para la copia de los archivos de Citlali a Andrómeda.

El telescopio RIS, es encendido manualmente y comienza su monitoreo a las 9:00 hrs cuando termina el monitoreo del sol, se apaga junto con los servidores Quetzal y Citlali cortando la comunicación con el servidor Andrómeda, por lo que se tomó la decisión de que cada mañana a las 9:00 hrs Andrómeda realice la copia de los archivos de un día anterior.

A continuación de muestra el primer algoritmo que se desarrolló para el programa `recepcionRis.sh`.

Algoritmo 1:

1. Pregunta si son las 9:00hrs.
2. Se abre una conexión ssh desde Andrómeda hacia el servidor Citlali, se hace el cálculo de la fecha del día anterior para conocer año, mes y día, con esos datos se coloca en la carpeta del servidor Citlali, /database/año/mes/día, en esta carpeta se encuentran los 4 archivos xdat que se generaron ese día.
3. Se ejecuta el comando: `scp -r "xitris@192.168.xxx.xxx: /database/año/mes/día/media/dataBase/ris/recibidos` para enviar todos los archivos de esta carpeta al servidor Andrómeda en la carpeta `/media/ dataBase/ris/recibidos`.
4. Una vez que termina la copia de todos los archivos generados se ejecuta

el programa `almacenaRis.jar`.

Cuando se puso en marcha este algoritmo, resultó claro que había dos fallas en el diseño, la primera fue que el RIS al ser encendido de manera manual, podía haber ocasiones en que no se encendiera exactamente a las 9:00 hrs teniendo un retraso de minutos o hasta un par de horas, el segundo error fue para el primer día de la semana. Dado que, el sábado y el domingo el RIS está apagado, entonces el sábado Andrómeda quiere copiar los archivos del viernes y no existe conexión para realizar la copia, mientras que el lunes el sistema preguntará por los datos del domingo y por lo tanto quedarán sin copiar los datos viernes, es posible hacer tan solo una variación para que el algoritmo se comporte diferente los lunes, sin embargo si existiera un día de asueto entre semana los datos del día anterior se perderían.

Para solucionar ese problema se desarrolló el algoritmo que se muestra a continuación:

Algoritmo 2:

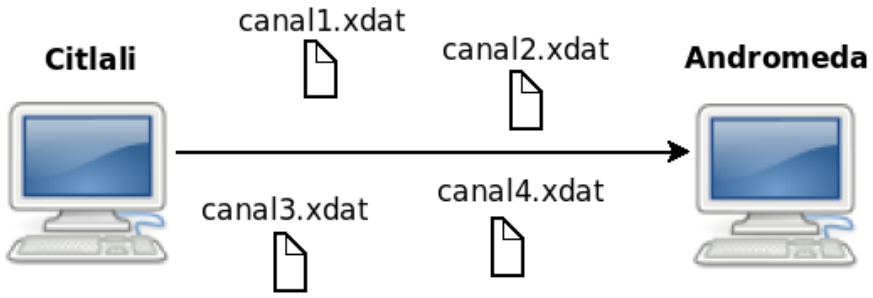
- mientras tiempo_1 == true, tiempo_1 es una bandera auxiliar para romper el while
- se calcula hora_actual
- si hora_actual == 9
- mientras tiempo_2 == true
  - si `ping` a RIS == true
    - se ejecuta la función `calculaAyer`
    - se calcula dia_ayer
    - se calcula la ruta de la carpeta del día de ayer en xitris: `/database/año/mes/dia`
    - con `ssh` se coloca en la ruta de la carpetas
    - con `scp` se hace la copia a la ruta: `/database/ris/recibidos` en el servidor Andrómeda de los 4 archivos `xdat`
    - se ejecuta `almacenaRis.jar`
    - se rompe el while tiempo_2 = false
  - si no
    - se duerme 3600 segundos
    - se pregunta la hora_actual2

  - si hora_actual2 == 14
  - se rompe el while tiempo_2 = false
- repetir
- se duerme 3600 segundos
- repetir

  Algoritmo de la función `calculaAyer`:

  - se hace una consulta a la base de datos Geofísica, que regresa el número de días que han pasado desde el último registro de la tabla `muestreoRis`, hasta el día de ayer.
  - se crean las fechas que hacen falta y con `scp` se realiza la copia desde xitris al servidor Andrómeda en una carpeta temporal: `/dababase/ris/temporal`.

Con este nuevo algoritmo quedaron resueltos los problemas para realizar las copias tanto en fines de semana como en días de asueto, el diagrama de flujo se muestra en la figura 3.3.



scp -r "xitris@192.168.xxx.xxx/database/año/mes/dia" /home/usuario/xitris/tmp
Este script esta dado de alta en Cron y se ejecuta todos los dias a las 8:00pm

Figura 3.4: Comando scp para el copiado de archivos de RIS a Andrómeda.

La figura 3.4 muestra el comando `scp`que se utilizó para realizar diariamente la copia de los archivos a Andrómeda.

Nota: `almacenaRis.jar` se encarga de tomar los `xdat` y convertirlos a un solo archivo FITS y subirlo a la base de datos, el algoritmo se explica en el capítulo 5.

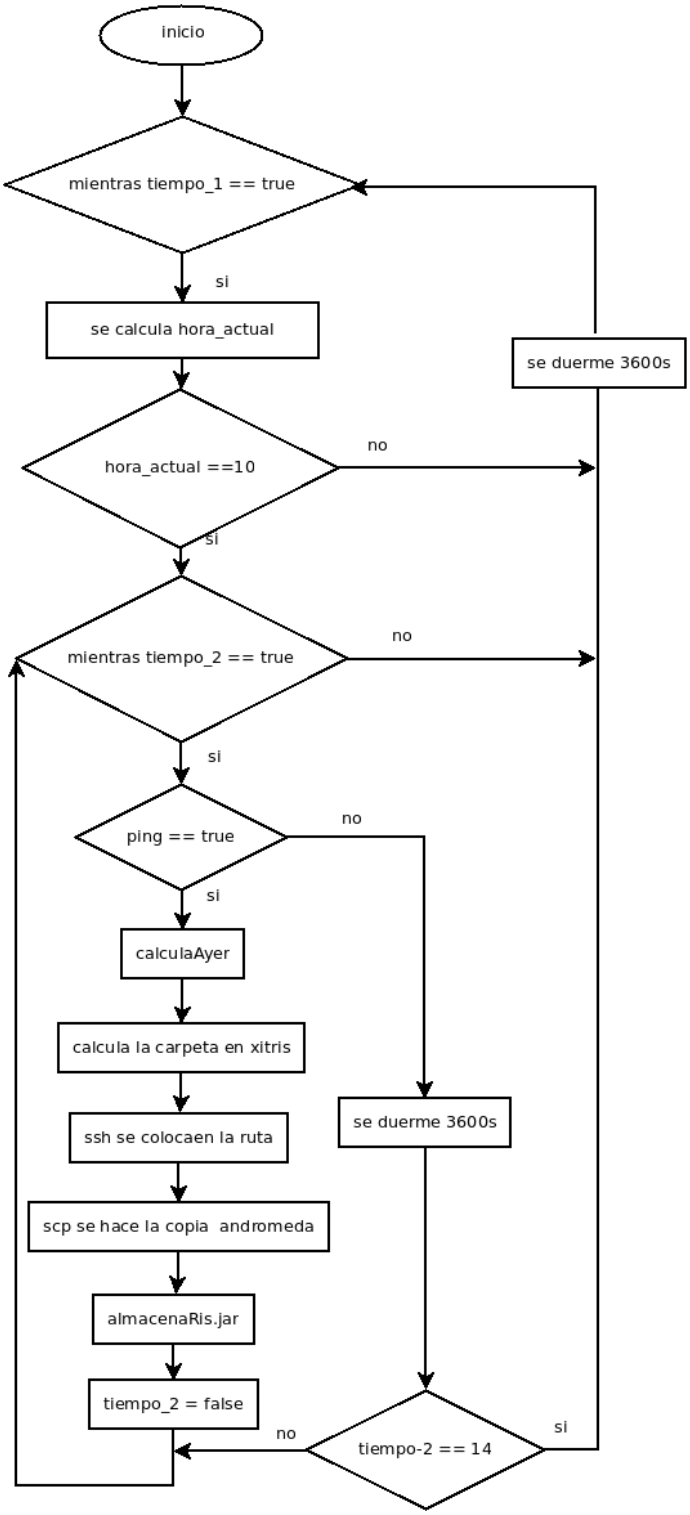


Figura 3.3: Diagrama de flujo para el programa recepcionRis.sh

3.3. Análisis del formato de salida de CALLISTO

El servidor CALLISTO siempre está encendido y genera archivos FITS cada 15 minutos, el tipo de archivo FITS que genera CALLISTO se encuentra en matrices de datos correspondientes a tiempo contra frecuencia, generando así un espectro dinámico.

Estos archivos son guardados en el servidor CALLISTO en la ruta `/Data`, el formato de su nombre es:

`UNAM_AñoMesDia_HoraMinutosSegundos_Milisegundos.fit`.

Estos archivos no pasarán por ningún proceso ya que su formato es FITS, en la figura 3.5 se muestra la cabecera de un archivo generado por CALLISTO.

```
SIMPLE  =                    T / file does conform to FITS standard
BITPIX  =                    8 / number of bits per data pixel
NAXIS   =                    2 / number of data axes
NAXIS1  =                 3600 / length of data axis 1
NAXIS2  =                  200 / length of data axis 2
EXTEND  =                    T / FITS dataset may contain extensions
COMMENT   FITS (Flexible Image Transport System) format is defined in 'Astronomy
COMMENT   and Astrophysics', volume 376, page 359; bibcode: 2001A&A...376..359H
COMMENT  File created by e-Callisto for Unix version 1.0.0
DATE    = '2015-03-30'         / Time of observation
CONTENT = '2015/03/30  Radio flux density, e-CALLISTO (UNAM)' / Title of image
ORIGIN  = 'Mexico_city'        / Organization name
TELESCOP= 'Radio Spectrometer' / Type of instrument
INSTRUME= 'UNAM     '          / Name of the spectrometer
OBJECT  = 'Sun      '          / object description
DATE-OBS= '2015/03/30'         / Date observation starts
TIME-OBS= '20:00:04.441'       / Time observation starts
DATE-END= '2015/03/30'         / date observation ends
TIME-END= '20:15:04'           / time observation ends
BZERO   =                   0. / scaling offset
BSCALE  =                   1. / scaling factor
BUNIT   = 'digits  '           / z-axis title
DATAMIN =                    0 / minimum element in image
--Más--(0%)
```

Figura 3.5: Cabecera de un archivo FITS generado por CALLISTO el 30 de marzo del 2015

3.4. Algoritmo para la recepción de los archivos del servidor CALLISTO

La recepción de los archivos FITS estará a cargo de un algoritmo programado en shell para el copiado de los archivos FITS por medio de `scp` desde CALLISTO a Andrómeda, este programa fue nombrado `recepcionCallisto.sh` y se

ejecutará todos los días a las 23:00h para garantizar que su periodo diario haya concluido, el algoritmo de este programa se describe a continuación.

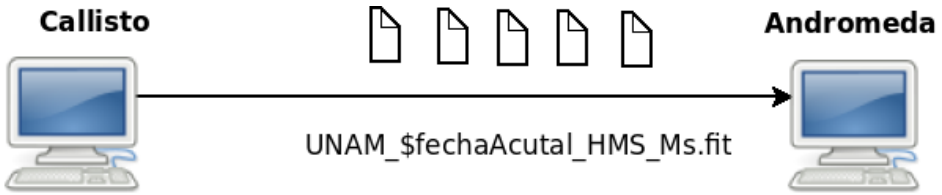
Algoritmo:

– Desde Andrómeda se abre una conexión `ssh` hacia el servidor CALLISTO, colocándose en su carpeta /Data, en este sitio se encuentran todos los archivos FITS generados.

– Calcula la fecha del día actual y se lista la carpeta buscando que el nombre del archivo contenga la cadena de la fecha del día actual. Para cada archivo encontrado se ejecuta el comando:
`scp "observer@132.248.xxx.xxx:Data/"$FechaActual/media/DataBase` para enviar los archivos FITS al servidor Andrómeda en la carpeta: `/media/DataBase/` .

– Una vez que se realizó la copia de todos los archivos encontrados se ejecuta el archivo `almacenaCallisto.jar` . Este programa se explicará en el capítulo siguiente ya que se encarga del almacenamiento en la base de datos Geofísica.

La figura 3.4 muestra el comando scp que se utilizó para realizar diariamente la copia de los archivos a Andrómeda.



scp -r "observer@132.248.xxx.xxx:Data"$fechaActual /media/DataBase/$fechaActual
Este script esta dado de alta en Cron y se ejecuta todos los dias a las 11:00pm

Figura 3.6: Comando scp para el copiado de archivos de CALLISTO a Andrómeda.

Capítulo 4

Base de datos.

El Sistema Andrómeda requiere de una base de datos que sea capaz de almacenar los datos de los archivos que generan los telescopios RIS y CALLISTO. Se trata de una base de datos principalmente de consulta e inserciones, donde diariamente se añadirán registros, estos registros no serán modificados, solamente serán consultados esencialmente por una interfaz web.

Una base de datos, es un conjunto de datos que se relacionan entre si, de manera estructurada, en atributos y tuplas.

Este conjunto de relaciones se llaman tablas, cada tabla cuenta con una llave primaria que es un atributo o varios, en el caso de una llave primaria compuesta, para identificar de forma única cada tupla.

La normalización de bases de datos fue introducida como procedimiento por Edgar Frank Codd, un científico de computación en IBM en su artículo "Un modelo relacional de datos para grandes bancos de datos compartidos" en 1970.

El proceso de normalización consiste en comprobar de manera secuencial si el esquema original está en la primera forma normal (1FN), segunda formal normal (2FN) y tercera forma normal (3FN) analizando las dependencias funcionales en cada paso.

El la tabla 4.1 se muestra una breve descripción de las reglas:

La aplicación de estas 3 formas normales ayuda en gran medida a:

- Reducir la redundancia en los datos.
- Evitar incongruencias en los datos.
- Proteger la integridad de los datos.

Forma Normal	Descripción
1FN	Suprime los grupos de repetición.
2FN	Dependencias parciales: Asegura que todos los atributos son dependientes de la llave completa.
3FN	Dependencias Transitivas: Asegura que los atributos que no son llaves, son independientes entre si, y dependientes solamente de la llave primaria.

Cuadro 4.1: Reglas de normalización

4.1. Análisis del escenario

RIS: El telescopio RIS cuenta con 4 canales de monitoreo, cada canal genera un archivo `xdat` por día.

CALLISTO: El telescopio CALLISTO genera un archivo FITS cada 15 minutos.

Los datos de interés son los siguientes:

RIS:

- archivo FITS generado
- hora en que se inició el monitoreo
- hora en que se terminó el monitoreo
- fecha de registro

CALLISTO:

- archivo FITS generado
- hora en que se inició el monitoreo
- hora en que se terminó el monitoreo
- fecha de registro

4.2. Análisis de requerimientos funcionales

– RIS: Los archivos `xdat` que genera el telescopio RIS serán procesados para crear un archivo FITS que contenga los datos de los 4 canales (este proceso se explicará en la sección 5.1), generando así un único archivo FITS por día, que deberá guardarse en una carpeta siguiendo la siguiente estructura: `año/mes/dia/archivo.fits`.

– CALLISTO: En un sólo día el telescopio CALLISTO genera 37 archivos FITS, cada archivo deberá guardarse en una carpeta siguiendo la siguiente estructura `año/mes/dia/archivo.fits`.

– Diariamente se deberán guardar los datos que haya generado cada telescopio, los datos guardados en la base de datos Geofísica no se modificarán a través del tiempo, sólo podrán ser consultados por medio de una interfaz web seleccionando la fecha en un calendario.

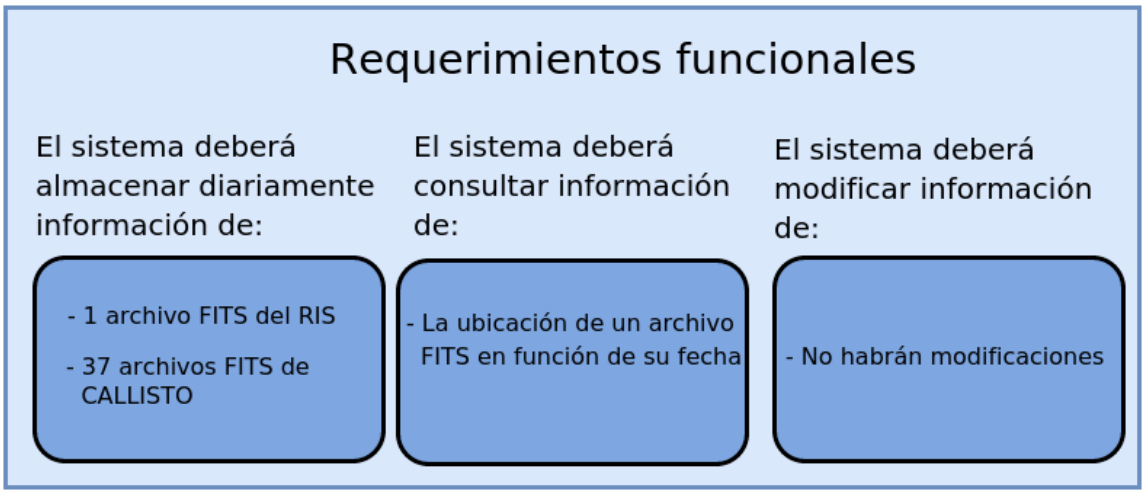


Figura 4.1: Requerimientos funcionales

4.3. Análisis de requerimientos no funcionales

Seguridad lógica y de datos:

– Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador de acceso a datos.

– Todas las comunicaciones externas entre los diferentes servidores, deben ser por medio del protocolo `ssh` .

– La base de datos no tendrá respaldos automáticos debido a restricciones de hardware, por lo que se harán a consideración del administrador en medios extraíbles.

Volumen RIS:

– Diariamente se genera un archivo FITS con un tamaño promedio de 138.20 KB.

– Anualmente se generan al rededor de 255 archivos, un promedio de 34.42 MB de información anualmente.

– Se cuentan con archivos desde el año 2004, acumulando 413.04 MB de información.

Volumen CALLISTO:

– Diariamente se genera 37 archivos FITS con un tamaño promedio de 760.30 KB cada uno, por lo que al día se generan 27.46 MB de información.

– Anualmente se generan al rededor de 13505 archivos, un promedio de 9.80 GB de información anualmente.

– Se cuentan con archivos desde el año 2014, acumulando 19.8 GB de información.

Usabilidad:

– El sistema debe contar con manual de usuario estructurado adecuadamente.

– El administrador del sistema no cuenta con conocimientos especializados en computación, por lo que el sistema se debe administrar de manera gráfica y amigable.

4.4. Sistema de gestión de bases de datos (RDBMS)

Para el diseño de la base de datos se utilizó el modelo de datos relacional, se optó por un manejador de base de datos de código libre por conveniencia del laboratorio. Esto redujo las posibilidades prácticamente a dos manejadores que

fueron PostgreSQL y MySQL.

El gestor de bases de datos que se utilizó fue MySQL, por requerimientos del sistema la página web esta diseñada sobre wordpress y esté trabaja con MySQL, para simplificar la administración del sistema Andrómeda se utilizó el mismo manejador para la base de datos Geofísica.

El sistema Andrómeda esta desarrollado por el conjunto de aplicaciones Apache-PHP-MySQL que es uno de los más utilizados para aplicaciones web.

4.5. Diseño de la base de datos relacional

Una decisión importante al momento del diseño, fue la manera de guardar los archivos FITS, MySQL cuenta con un tipo de datos llamado: LONGBLOB, este tipo de datos binario puede contener una cantidad variable de datos y permite almacenar archivos con tamaños de hasta 4.294.967.295 bytes.

Esta manera de almacenar los archivos permite que éstos se traten como un tipo de dato primitivo más de la base de datos. La segunda opción sería guardar los archivos en carpetas y hacer referencia a ellos por sus rutas absolutas como un registro en la base de datos. Una tercera opción es guardar las matrices de datos de cada archivo FITS en la base de datos y cada que se busque un archivo generarlo en el momento.

En cuanto al diseño para la base de datos Geofísica, se eligió la segunda opción, ya que entre los requerimientos se especifica que los archivos deben estar en carpetas. El inconveniente en este diseño es que al incrementar el número de archivos, la administración de las carpetas y archivos es más compleja, en caso de algún ataque los archivos están en claro y pueden ser modificados.

Una vez definidos estos puntos se realizó el análisis de las entidades. En bases de datos, una entidad representa a un objeto ya sea real o abstracto, que representa interés para el sistema y del cual se obtiene información.

Estas entidades cuentan con atributos que son los datos que definen o identifican a esta entidad. Deben contar al menos con un identificador único llamado llave primaria (PK) y si es necesario una llave foránea (FK), que a su vez es el identificador único de otra entidad con la finalidad de relacionarlas entre si, siempre y cuando exista

una dependencia entre ellas.

Contamos con dos entidades principales para la base de datos Geofísica las cuales son: `muestreoRis` y `muestreoCallisto` estas entidades tienen como atributos los datos del muestreo diario y las rutas de sus archivos FITS. Estas entidades se relacionan entre si por una tercera entidad llamada instrumento que tiene como atributos las especificaciones técnicas de RIS y CALLISTO.

Una primera aproximación de las entidades con sus atributos quedaría de la siguiente manera:

- `instrumento(id_instrumento(PK), ficha_técnica, fecha_adquisición, origen)`
- `muestreoRis(id_muestreo(PK),fecha, hora_inicio, hora_fin, ruta_fits, instrumento(FK))`
- `muestreoCallisto(id_muestreo(PK),fecha, hora_inicio, hora_fin, ruta_fits, instrumento(FK))`
- `xitris(id_axis(PK), canal1, canal2, canal3)`

`instrumento`: esta tabla contiene la información de cada telescopio, por el momento solo tendremos dos registros el de RIS y el de CALLISTO, sin embargo la base de datos esta diseñada para que se pueda expandir y guardar los registros de otros telescopios si fuera necesario.

`muestreoRis`: aquí se guardan los datos del muestreo de cada día del telescopio RIS como son, la fecha, la hora en la que se inicio y la hora en que de término el muestreo, la ruta donde está guardado su archivo FITS, un identificador único llamado id_muestreo y una llave foránea para identificar que es un registro del telescopio RIS.

`muestreoCallisto`: esta tabla contiene los datos del muestreo de CALLISTO, la fecha, la hora en la que se inició y la hora en que terminó el muestreo, la ruta donde esta guardado su archivo FITS, un identificador único llamado id_muestreo y una llave foránea para identificar que es un registro del telescopio CALLISTO.

`xitris`: es una tabla auxiliar temporal para la creación del archivos FITS, proceso que se explica en la sección 3.1

4.6. Propuesta y Normalización del Diagrama Entidad Relación (DER)

Para el modelado de la Base de Datos Geofísica, se utilizó el Modelo Relacional. Este diagrama es una representación gráfica de las entidades o tablas de la base de datos, muestra sus atributos y sus relaciones con otras entidades.

La figura 4.2 muestra la primer propuesta del DER, donde se retoman las cuatro tablas descritas anteriormente :

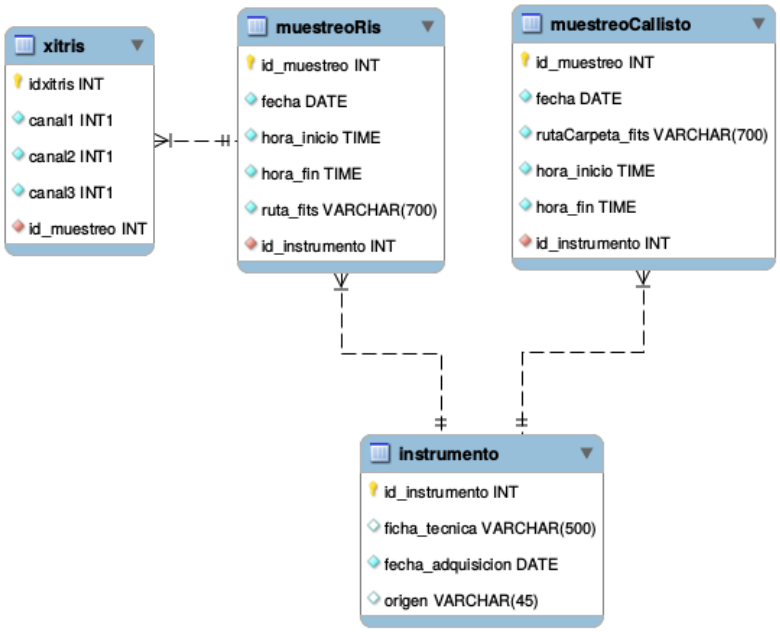


Figura 4.2: Primer propuesta del DER

Para asegurar la integridad y la consistencia de los datos, se aplicarán las reglas de normalización.

A continuación se aplica la primera forma normal para asegurar que entre los campos de las tablas no existan grupos de repetición.

1ra Forma normal: En la figura 4.3 se muestra la tabla `muestreoCallisto` con algunas tuplas ejemplo, en los atributos `id_muestreo`, `id_instrumento` y `fecha` se están repitiendo registros, debido a que CALLISTO genera aproximadamente 37 archivos FITS por día, la tabla viola la 1ra forma normal

que hace referencia a la atomicidad de los registros, es decir no se deben tener grupos repetidos, en este caso para el `id_muestreo = 1` se están generando varios registros por lo que se tuvo que crear una tabla nueva llamada `archivosCallisto` con la finalidad de agrupar los registros repetidos para cada muestreo por día.

muestreoCallisto

id_muestreo	id_instrumento	fecha	hora_inicio	hora_fin	ruta_fits
1	2	2011-06-03	21:00:02	21:15:02	/media/dataBase/callisto/2011/06/03/UNAM_20110603_210002_59.fit
1	2	2011-06-03	20:30:02	20:45:02	/media/dataBase/callisto/2011/06/03/UNAM_20110603_203002_59.fit
1	2	2011-06-03	21:30:02	21:45:02	/media/dataBase/callisto/2011/06/03/UNAM_20110603_213002_59.fit
1	2	2011-06-03	19:15:01	19:30:01	/media/dataBase/callisto/2011/06/03/UNAM_20110603_191501_59.fit
1	2	2011-06-03	19:45:02	18:00:02	/media/dataBase/callisto/2011/06/03/UNAM_20110603_194502_59.fit
1	2	2011-06-03	20:45:02	21:00:02	/media/dataBase/callisto/2011/06/03/UNAM_20110603_204502_59.fit
1	2	2011-06-03	22:30:02	22:45:02	/media/dataBase/callisto/2011/06/03/UNAM_20110603_223002_59.fit
1	2	2011-06-03	18:00:01	18:15:01	/media/dataBase/callisto/2011/06/03/UNAM_20110603_180001_59.fit
1	2	2011-06-03	16:45:01	17:00:01	/media/dataBase/callisto/2011/06/03/UNAM_20110603_164501_59.fit
1	2	2011-06-03	21:45:02	22:00:02	/media/dataBase/callisto/2011/06/03/UNAM_20110603_214502_59.fit

Figura 4.3: Tabla muestreoCallisto

En la figura 4.4 se muestra la segunda propuesta del DER que incluye la tabla `archivosCallisto` :

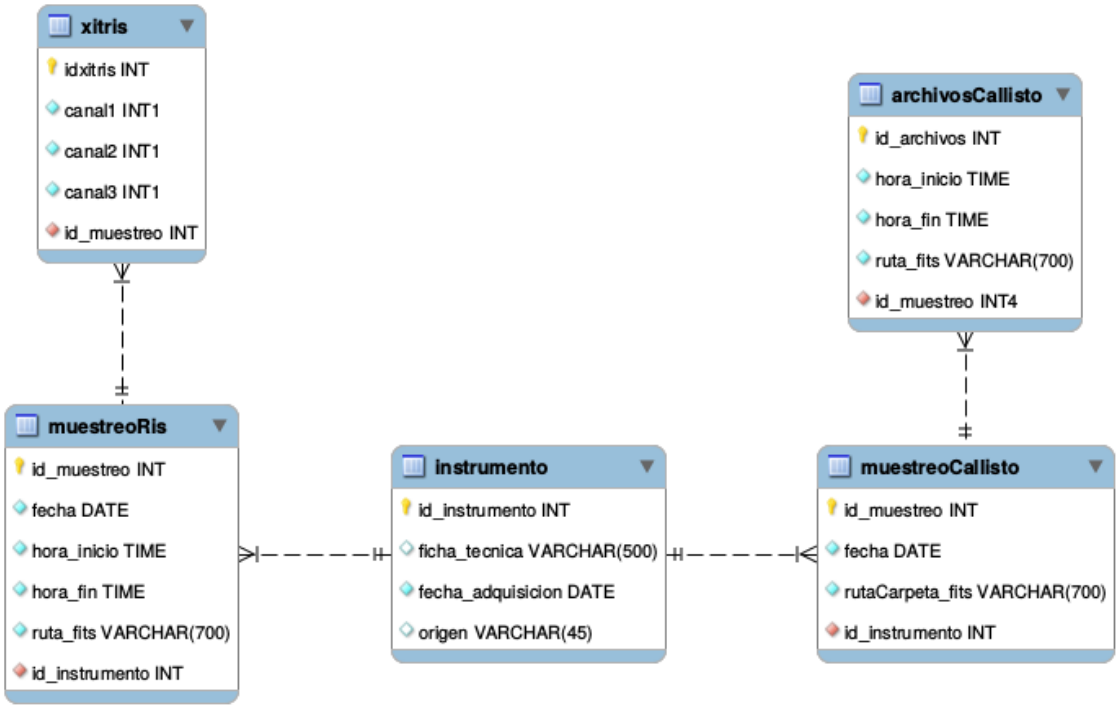


Figura 4.4: Segunda propuesta del DER

La tabla `archivosCallisto` tiene los datos de los 37 archivos para cada muestreo por día que toma CALLISTO como son, la hora de inicio, la hora de finalización y la ruta para cada archivo FITS, también cuenta con una llave foránea de la tabla `muestreoCallisto` para agrupar los archivos que se generaron en el muestreo por día.

A continuación se muestran las tablas candidatas a las cuales se les aplicarán las tres formas normales:

muestreoCallisto

id_muestreo	fecha	rutaCarpeta_fits	id_instrumento
1	2011-06-03	/media/dataBase/callisto/2011/06/03/	2

Figura 4.5: Tabla muestreoCallisto

archivosCallisto

id_archivo	hora_inicio	hora_fin	ruta_fits	id_muestreo
1	21:00:02	21:15:02	/media/dataBase/callisto/2011/06/03/UNAM_20110603_210002_59.fit	1
2	20:30:02	20:45:02	/media/dataBase/callisto/2011/06/03/UNAM_20110603_203002_59.fit	1
3	21:30:02	21:45:02	/media/dataBase/callisto/2011/06/03/UNAM_20110603_213002_59.fit	1
4	19:15:01	19:30:01	/media/dataBase/callisto/2011/06/03/UNAM_20110603_191501_59.fit	1
5	19:45:02	18:00:02	/media/dataBase/callisto/2011/06/03/UNAM_20110603_194502_59.fit	1
6	20:45:02	21:00:02	/media/dataBase/callisto/2011/06/03/UNAM_20110603_204502_59.fit	1
7	22:30:02	22:45:02	/media/dataBase/callisto/2011/06/03/UNAM_20110603_223002_59.fit	1
8	18:00:01	18:15:01	/media/dataBase/callisto/2011/06/03/UNAM_20110603_180001_59.fit	1
9	16:45:01	17:00:01	/media/dataBase/callisto/2011/06/03/UNAM_20110603_164501_59.fit	1
10	21:45:02	22:00:02	/media/dataBase/callisto/2011/06/03/UNAM_20110603_214502_59.fit	1

Figura 4.6: Tabla archivosCallisto

Instrumento

id_instrumento	ficha_tecnica	fecha_adquisición	origen
1	Radio Interferómetro Solar de base pequeña del Instituto de Geofísica (RIS)		
2	Espectrómetro solar CALLISTO		Suiza

Figura 4.7: Tabla instrumento

muestreoRis

id_muestreo	fecha	hora_inicio	hora_fin	ruta_fits	id_instrumento
1	2004-10-01	13:18:32	20:00:06	/media/dataBase/ris/2004/10/01/Risc2004-10-01.fits	1
2	2004-09-02	16:17:08	22:31:44	/media/dataBase/ris/2004/09/02/Risc2004-09-02.fits	1
3	2004-08-13	16:17:08	22:31:44	/media/dataBase/ris/2004/08/13/Risc2004-08-13.fits	1
4	2004-08-26	16:17:08	22:31:44	/media/dataBase/ris/2004/08/26/Risc2004-08-26.fits	1
5	2004-12-20	21:08:05	21:11:23	/media/dataBase/ris/2004/12/20/Risc2004-12-20.fits	1
6	2004-04-27	21:08:05	21:11:23	/media/dataBase/ris/2004/04/27/Risc2004-04-27.fits	1
7	2004-10-14	14:29:12	22:51:06	/media/dataBase/ris/2004/10/14/Risc2004-10-14.fits	1
8	2012-03-23	14:14:03	21:25:18	/media/dataBase/ris/2012/03/23/Risc2012-03-23.fits	1
9	2005-08-05	14:29:37	00:14:11	/media/dataBase/ris/2005/08/05/Risc2005-08-05.fits	1
10	2008-09-17	13:57:07	00:03:50	/media/dataBase/ris/2008/09/17/Risc2008-09-17.fits	1

Figura 4.8: Tabla muestreoRis

xitris

id_xitris	canal1	canal2	canal3
1	500	300	0
2	452	352	52
3	520	320	24
4	256	356	52
5	542	342	60
6	652	352	12
7	542	342	25
8	254	354	95
9	421	321	32
10	520	320	52

Figura 4.9: Tabla xitris

1ra Forma normal: Todas las tablas cumplen con el principio de atomicidad en los registros, se puede observar que en ninguna tabla propuesta existen grupos repetidos.

2da Forma normal: La segunda forma normal trata de las dependencias parciales, el DER propuesto cumple con esta forma normal, ya que cada atributo depende de la llave completa en cada tabla.

En la tabla `muestreoRis`, todos los atributos corresponden al muestreo que se hace por día del telescopio RIS, en la tabla `muestreoCallisto`, todos los atributos dependen del muestreo que se hace diariamente de CALLISTO, para la tabla `archivosCallisto`, todos sus atributos dependen de cada archivo generado, lo mismo pasa para la tabla instrumento, que sólo cuenta con atributos que dependen de cada instrumento y la tabla `xitris` que es una tabla auxiliar para crear los archivos FITS, sus atributos sólo dependen de su llave para crear un archivo.

3ra Forma normal: La tercera forma normal se refiere a la dependencia transitiva: es decir, todos los atributos de la tabla deben depender solamente de la llave primaria, el DER propuesto cumple con la 3ra forma normal ya que sus atributos son independientes entre si, pero cada uno depende solamente de la llave primaria.

Por lo tanto el DER de la segunda propuesta (ver figura 4.4) es con el que se trabajó ya que cumple con las reglas de normalización.

4.7. Rendimiento de la base de datos Geofísica

El sistema Andrómeda cuenta con un histórico de datos, en la sección 5 se explica la carga de éstos. Una vez que la base de datos cuenta con todos los registros, se hace un análisis sobre las consultas fijas, estas búsquedas se realizan desde una página web por medio de un calendario por lo que la búsqueda se hace en función a la fecha en las tablas `muestreoRis` y `muestreoCallisto` .

Para la tabla `muestreoRis` la consulta se realiza con un tiempo de 0.0012 segundos, como podemos ver en la imagen 4.10. Esta consulta es recurrente por lo que es conveniente crear un índice para el atributo fecha y así optimizar el tiempo de búsqueda.

Figura 4.10: Búsqueda sin índice en la tabla muestreoRis

Una vez que el índice fue implementado, la reducción del tiempo de respuesta de la búsqueda fue de 0.0012 a 0.0002 segundos. La consulta se puede observar en la

imagen 4.11.

Figura 4.11: Búsqueda con índice en la tabla muestreoRis

En la tabla `muestreoCallisto` la consulta se realiza con un tiempo de 0.0003 segundos, como podemos ver en la imagen 4.12. Esta consulta al igual que la de `muestreoRis` es recurrente por lo que es conveniente crear un índice para el atributo fecha y así optimizar el tiempo de búsqueda.

Figura 4.12: Búsqueda sin índice en la tabla muestreoCallisto

Cuando el índice fue implementado la reducción del tiempo de respuesta de la búsqueda fue un éxito, el tiempo se redujo de 0.0003 a 0.0002 segundos. La consulta se puede observar en la imagen 4.13.

Figura 4.13: Búsqueda con índice en la tabla muestreoCallisto

La tabla `muestreoRis` cuenta con 1751 registros, el espacio total utilizado incluyendo los índices para estos registros es de, 158.1 KB de información como se muestra en la imagen 4.14, estos registros son del 3 de marzo del 2004 al 8 de agosto del 2016.

Figura 4.14: Datos de la tabla muestreoRis

La tabla `muestreoCallisto` cuenta con 1294 registros y el espacio total utilizado incluyendo los índices para estos registros es de, 105.7 KB de información como se muestra en la imagen 4.15, estos registros son del 29 de abril del 2011 al 16 de agosto del 2016.

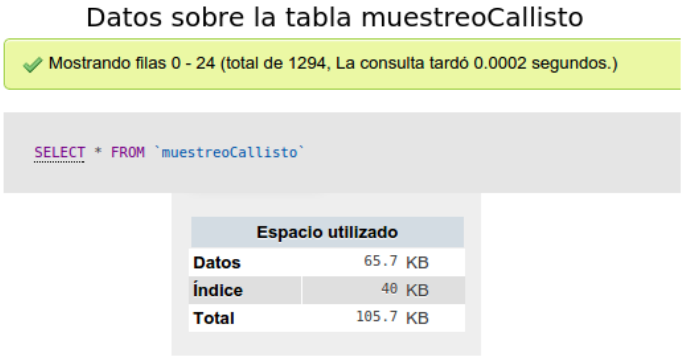


Figura 4.15: Datos de la tabla muestreoCallisto

La base de datos Geofísica al ser una base de datos histórica cuenta con una gran cantidad de registros que se están incrementando diariamente, una mejora en el diseño es hacer un estudio de cuales son los periodos más solicitados y crear particiones para mejorar el rendimiento.

Capítulo 5

Carga de datos en la BD Geofísica.

En el capítulo anterior se describió el diseño de la base de datos para guardar diariamente la información obtenida por los telescopios, en este capítulo se mostrará el proceso de transformación y de carga. En un principio la idea fue programar todo el Sistema Andrómeda en el lenguaje de programación Java, los archivos ejecutables de Java son archivos `jar`, listos para ejecutarse en cualquier sistema operativo que cuente con la maquina virtual de Java.

Para crear los archivos que cumplen con el estándar FITS, la página oficial de la NASA: `http://fits.gsfc.nasa.gov/fits_libraries.html`, sugiere una serie de bibliotecas en diferentes lenguajes de programación, tres de ellas son de Java.

Cuando se implementaron las bibliotecas de java para crear los archivos FITS para el telescopio RIS, se hicieron pruebas de lectura con los programas que se utilizan en el laboratorio, estas lecturas no arrojaban los datos esperados, una posible causa es que los programas no son compatibles con esas bibliotecas de java que se probaron, ya que los archivos fueron verificados en un portal oficial: `http://fits.gsfc.nasa.gov/fits_verify.html` y no se presentaron errores (ver figura 5.1), bajo este escenario se continuo probando con otras bibliotecas.

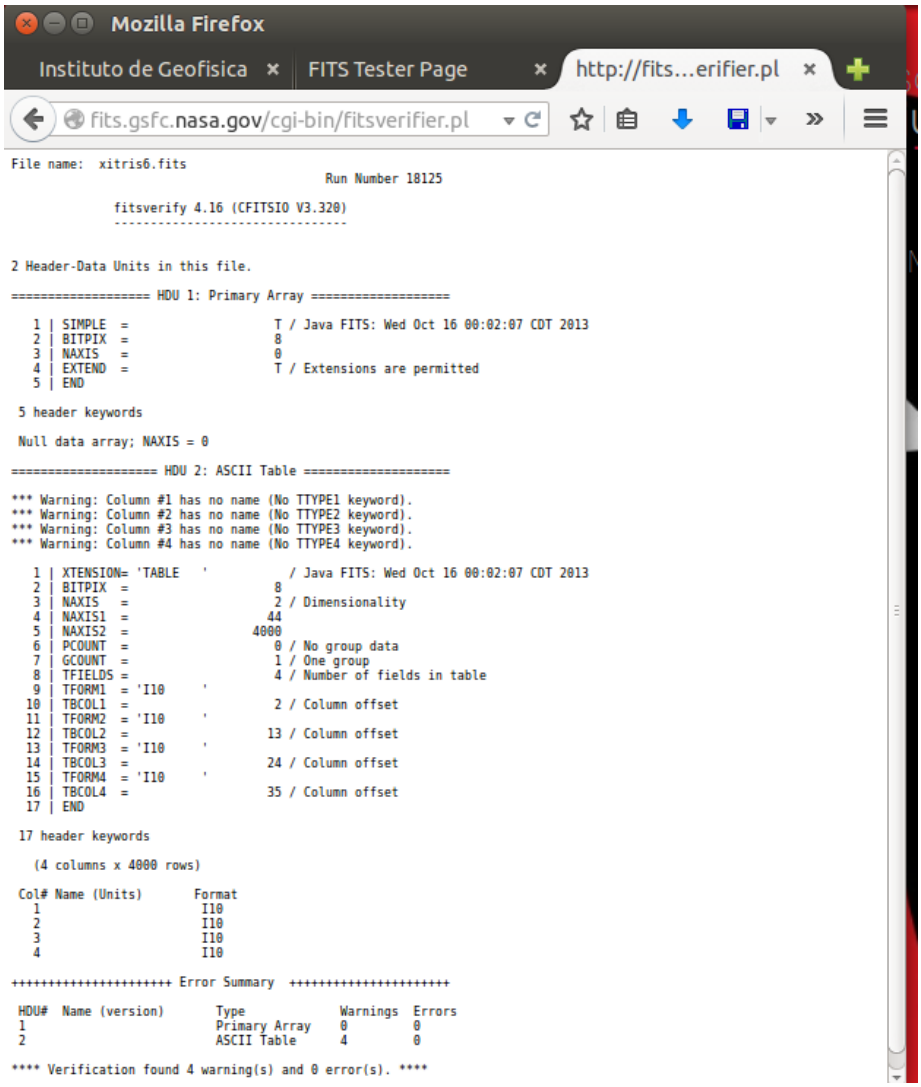


Figura 5.1: Resultado de la prueba del archivo Fits generado con el lenguaje Java

Entre los lenguajes que propone el portal de la NASA esta Python, este lenguaje es de gran uso en la astronomía ya que cuenta con una serie de bibliotecas dedicado al análisis de datos en estas áreas.

La biblioteca que se probó fue PyFITS, se generó un archivo FITS que fue verificado correctamente en el portal (ver figura 5.2) y que fue cargado correctamente por los programas con los que se trabaja en el laboratorio. Por lo que la generación de los archivos FITS del telescopio RIS fue desarrollado en Python con la biblioteca PyFITS.

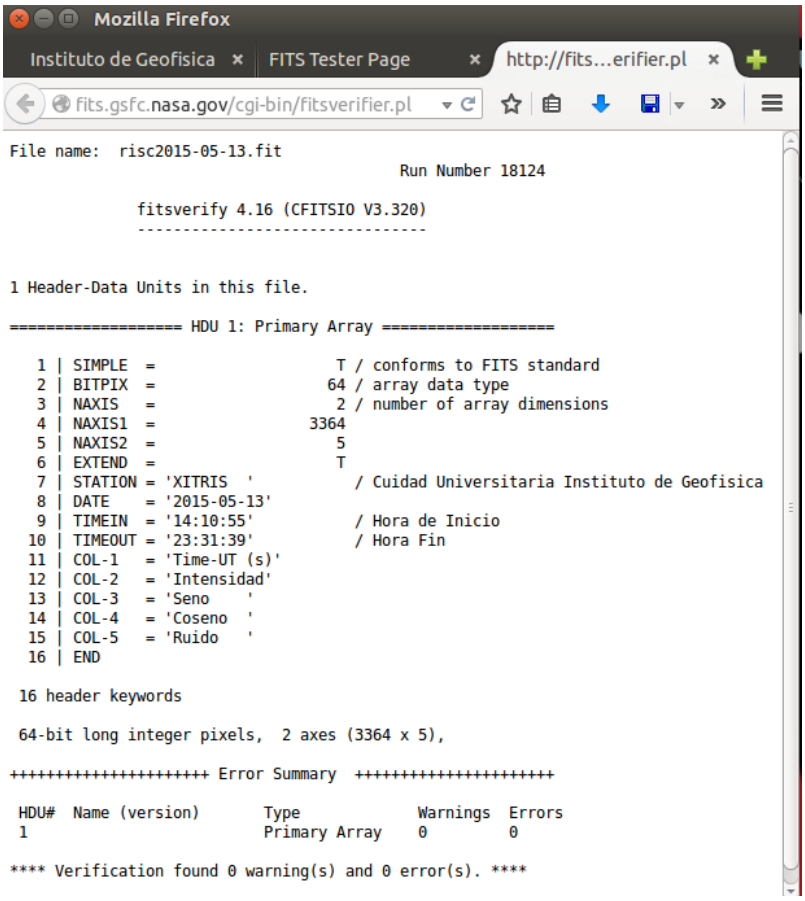


Figura 5.2: Resultado de la prueba del archivo Fits generado con el lenguaje Python

5.1. Algoritmo para la recepción diaria de los archivos del servidor RIS

– **almacenaRis.jar:** Este es un programa en lenguaje java que se encarga de tomar cada día los archivos que por medio de `scp` se copiaron del servidor Citlali a la carpeta `/media/dataBase/ris/recibidos` del servidor Andrómeda (esta etapa se explicó a detalle en el capítulo 3.2).

El programa buscará los cuatro archivos `xdat` que corresponden a cada canal del telescopio RIS, los convertirá con ayuda de `xdat2txt` a archivos temporales con los datos descifrados para guardarlos en la tabla xitris, una vez que haya terminado con los 4 archivos ejecutará el programa `hacerFits.py` para generar el archivo

FITS como una tabla de datos y guardará los datos del muestreo y la ruta del archivo FITS en la tabla `muestreoRis`.

El programa `xdat2txt`, se encarga de descifrar los archivos `xdat`, está escrito en lenguaje C y fue desarrollado en el laboratorio (para mayor información: `http://132.248.9.195/ptd2005/00327/0347735/Index.html`).

Una vez que el archivo está descifrado, se debe hacer un filtrado de los datos específicos que estamos buscando como son: `hora_inicio`, `hora_fin`, `fecha`, para realizar estas búsquedas se utiliza el comando `egrep`, este es un comando en el shell de linux que nos permite realizar búsquedas de patrones dentro de los archivos.

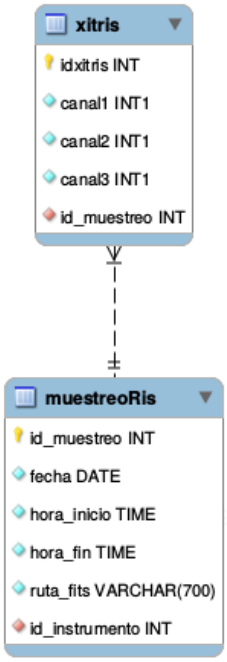


Figura 5.3: Tablas involucradas para el RIS

Algoritmo del programa `almacenaRis.jar`:

- Se calcula la ruta y se lista la carpeta.
- Si es un `xdat` entonces entra al método convertir.
  - `CONVERTIR`: Con el programa `xdat2txt` se genera un archivo llamado fecha.tmp donde están todos los datos descifrados y se pasan por un filtro.
  - `FILTRO`: Este filtro realiza un `egrep` donde se recolectan los datos como son, la hora, fecha etc.

  - Se guardan los datos generales, en la tabla `muestreoRis` de la base de datos.
  - se calcula la ruta de la carpeta del día anterior en xitris: `/database/año/mes/dia`
  - Se guardan los datos descifrados en la tabla xitris (xitris es una tabla temporal para crear el archivo FITS solamente).
- Se ejecuta el programa `crearFits.py`, que se encarga de generar el archivo FITS y guardarlo en `/media/dataBase/ris/año/mes/dia`.
- Se borra la tabla temporal xitris.

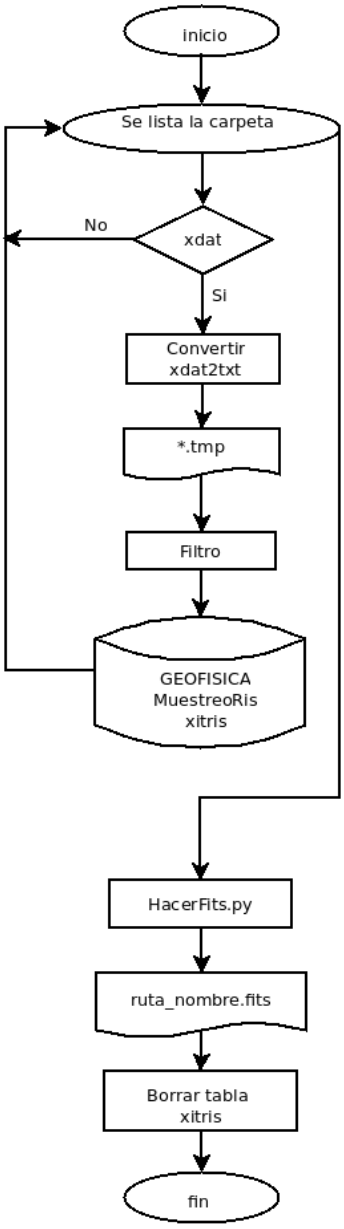


Figura 5.4: Diagrama de flujo del programa almacenaRis.jar

Algoritmo del programa crearFits.py:

- Recibe como parámetro la ruta del archivo FITS donde se debe guardar el archivo.
- Se realiza una conexión a la base de datos para consultar la `fecha`, `hora_inicio`, `hora_fin` en la tabla `muestreoRis` que corresponden al

archivo de la ruta FITS.

- Se consulta la tabla temporal xitris para guardar en un arreglo los datos del `canal_1`, `canal_2`, `canal_3` y `canal_4`.
- Con los datos recuperados por medio de la biblioteca pyfits se genera la cabecera del archivo FITS, se guarda la tabla de datos de los canales como cuerpo del FITS.
- Se guarda en la ruta que se envió como parámetro.

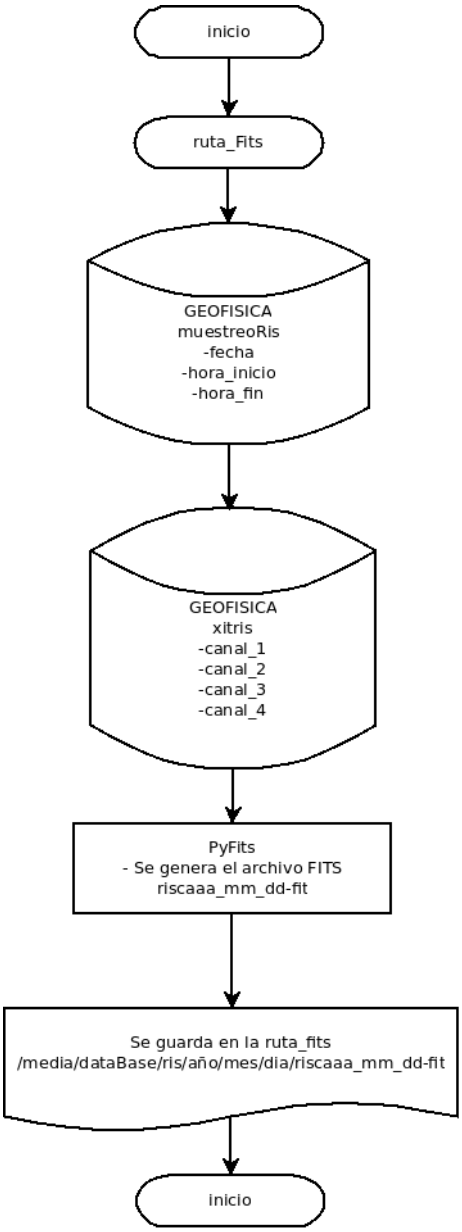


Figura 5.5: Diagrama de flujo del programa crearFits.py

5.2. Carga de datos históricos para el telescopio RIS

El telescopio RIS llegó al instituto en los 80's, desde ese tiempo comenzó su funcionamiento, y los datos se han ido almacenando en el servidor Quetzal desde el año 2004 con el siguiente formato `/database/año/mes/dia/canalX.xdat`, en este trabajo se retoman estos datos para cargarlos a la base de datos Geofisica y ponerlos a la disposición de manera online al público en general.

Se diseñó un algoritmo que fue programado en java y nombrado `historicoRis.jar` para almacenar estos datos y que puedan ser visualizados junto con los que se van generando diariamente.

Se creó un programa llamado `almacenaRis.jar` (ver capitulo 5.1) que se encarga de tomar los cuatro archivos xdat y meterlos a la base de datos. Tomando en cuenta lo anterior, el algoritmo `historicoRis.jar` selecciona de los archivos históricos los cuatro archivos que corresponden a un muestreo de RIS y los copia a la carpeta de trabajo de `almacenaRis.jar` para que los vaya guardando tal y como si se tratará del muestreo de un día común, esto lo hace recursivamente hasta terminar con todos los archivos.

5.3. Inserción de datos para el telescopio CALLISTO día a día

- **almacenaCallisto.jar:** Realiza la inserción de los datos del muestreo diario en la base de datos geofísica y copia los archivos en las rutas calculadas de acuerdo a su fecha (el proceso fue desarrollado en el capitulo 2.2). Este programa lo manda a ejecutar `recepcionCallisto.sh` una vez que termina de copiar los archivos en `/media/DataBase/callisto`.

Estructura del nombre: `UNAM_aaaammdd_hhmmss_ms.fit`.

Algoritmo del programa `almacenaCallisto.jar`:

- Se lista la carpeta CALLISTO.
  - Se toman los archivos FITS.
  - Se toma del nombre su fecha y su hora, para generar sus rutas.

  - Se toma cada archivo y se guardan sus datos en la tabla archivosCallisto.
- Una vez terminado el proceso con todos los archivos se guardan los datos correspondientes en la tabla `muestreoCallisto`.

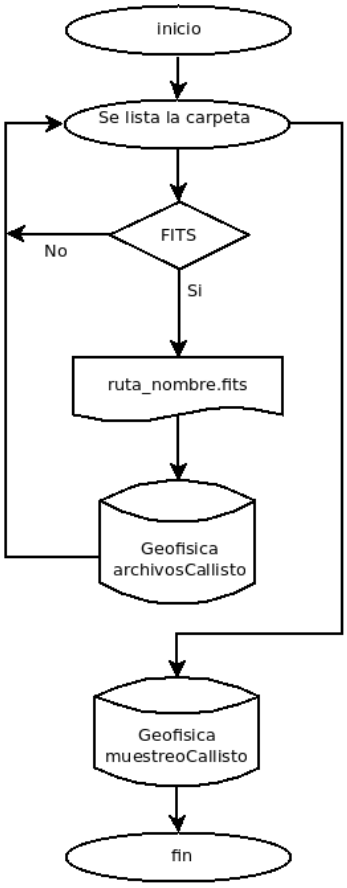


Figura 5.6: Diagrama de flujo del programa almacenaCallisto.jar

5.4. Carga de datos históricos para el telescopio CALLISTO

- **historicoCallisto.jar:** Realizará la inserción de los datos del muestreo histórico en la base de datos geofísica y copia los archivos en las rutas calculadas de acuerdo a su fecha. Todos los archivos que CALLISTO ha generado se guardan en la carpeta `/media/database` y se ejecuta el programa.

Algoritmo del programa `historicoCallisto.jar`:

- Se lista la carpeta CALLISTO.
  - Se toman los archivos FITS.
  - Se toma del nombre su fecha y su hora, para generar sus rutas.
  - Se toma cada archivo y se guardan sus datos en la tabla `archivosCallisto`.
- Una vez terminado el proceso con todos los archivos se guardan los datos correspondientes en la tabla `muestreoCallisto`.
- Se lista la carpeta `/media/databasey` del primer archivo se toma la fecha de su nombre.
- Se buscan todos los archivos con esa fecha y se copian a: `/media/DataBase/callisto`.
- Se repite el proceso desde el inicio.

Capítulo 6

Visualización.

6.1. Diseño de interfaz web

Hasta esta etapa del proyecto los datos históricos de los telescopios ya están guardados en la Base de Datos y la información ya se almacena diariamente, es momento de ocuparse en la visualización. Por los requerimientos y su fácil mantenimiento el portal principal fue elaborado con un sistema de gestión de contenido (CMS: Content Management System) que consiste en una interfaz totalmente gráfica que es administrada por una base de datos, esto permite crear una estructura para la página web haciendo que sus cambios sean de manera gráfica mediante una interfaz web.

Para el Sistema Andrómeda se utilizó WordPress, éste CMS es de código libre bajo la licencia GPL por lo que se adapta perfectamente a las necesidades del portal.

Se decidió con base a su sencillez y su estructura "Generate Press" ver figura 6.1. Se puede descargar de su página oficial `http://generatepress.com/`.

Este portal es el principal del Laboratorio de El Radio Observatorio Solar del Instituto de Geofísica de la UNAM, por lo que deberá tener el contenido general del laboratorio. Una vez que se estructuró el contenido que es necesario mostrar, el diseño final se muestra en la siguiente figura 6.2 .

Para RIS se mostrarán los datos en una gráfica con los tres canales que están en funcionamiento, se podrá descargar el archivo FITS que corresponde al día seleccionado ver figura 6.3.

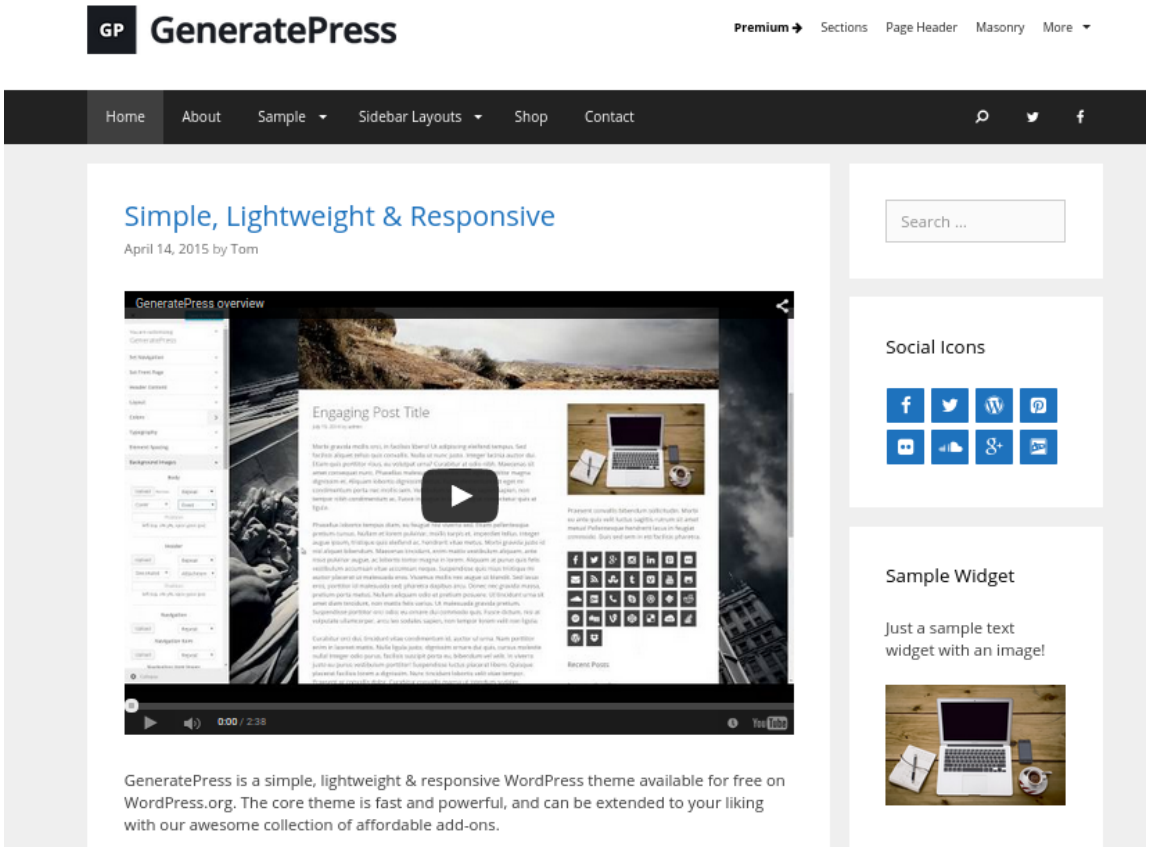


Figura 6.1: Plantilla Generate Press

Figura 6.2: Portal web de El Radio Observatorio Solar del Instituto de Geofísica de la UNAM

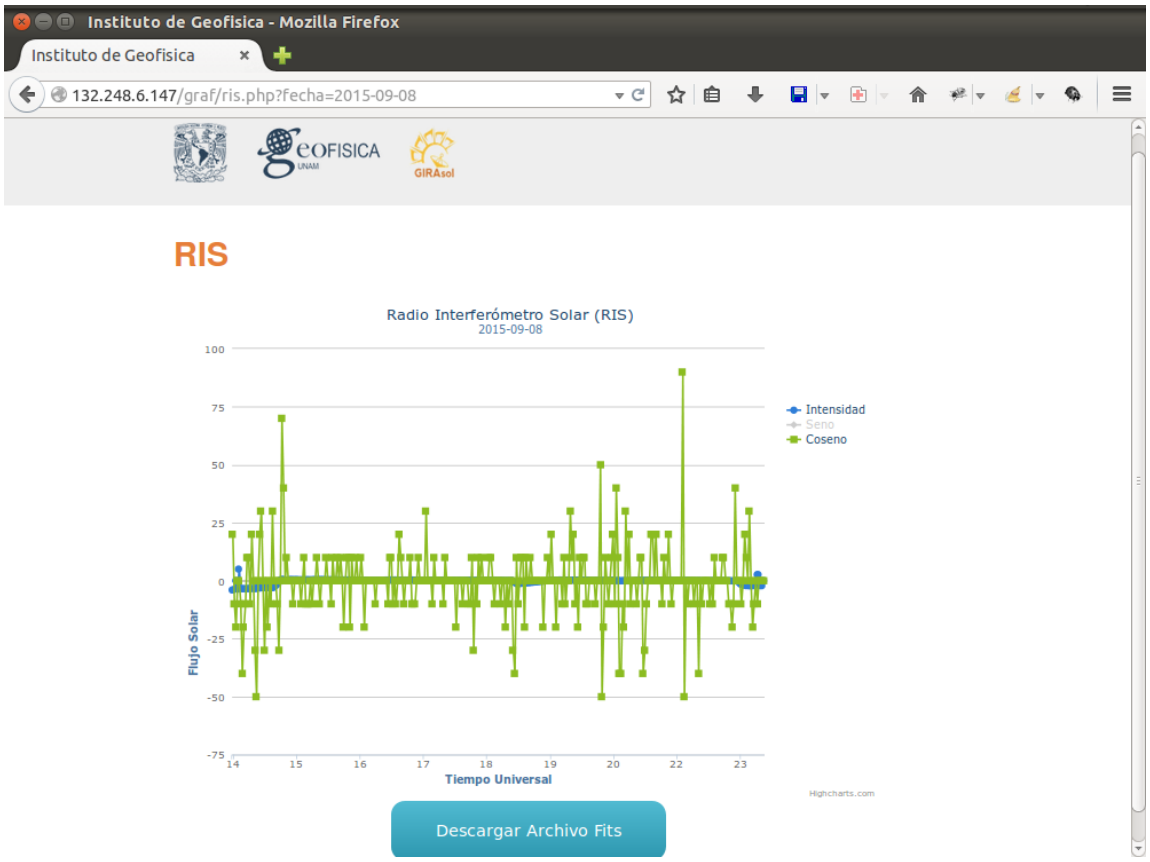


Figura 6.3: Página web del RIS

Para CALLISTO se mostrará una cuadricula de las imágenes generadas del día seleccionado y se podrá descargar un archivo `zip` que contendrá el conjunto de archivos FITS.

Para la visualización de los datos, fue conveniente programarla independientemente del manejador de contenido ya que requiere de una programación más especializada que involucra la consulta a la base de datos Geofísica y una serie de bibliotecas para el manejo de los archivos FITS.

Los lenguajes y bibliotecas utilizadas para esta parte son:

- **PHP:** es un potente lenguaje de programación para desarrollo web que se ejecuta del lado del servidor, por medio de este tendremos las consultas a la base de datos.
- **JQUERY:** es una biblioteca de java script que se utilizó para recoger los

Figura 6.4: Página web de CALLISTO

datos recopilados por PHP y mostrarlos en la página web por medio de AJAX.

– **AJAX:** es una técnica para comunicarse de manera asíncrona con el servidor permitiendo realizar cambios sobre las páginas sin necesidad de recargarlas.

– **HighCharts:** es una biblioteca de gráficos escritos en JavaScript, es utilizado para la gráfica de RIS.

– **CSS3:** es un lenguaje que se encarga de dar estilo a la página web, con esto le daremos diseño y presentación.

– **Python:** es un lenguaje de programación que se utilizó para la creación de las imágenes de CALLISTO.

Como resultado de este trabajo, el portal está a la vanguardia con las nuevas tecnologías por lo que es totalmente responsivo, es decir se adapta y visualiza automáticamente en cualquier dispositivo móvil.

6.2. Diseño del algoritmo de Visualización para RIS de los datos históricos

La visualización de los datos del RIS se lleva a cabo de la siguiente forma:

– Entra a la página http://cintli.geofisica.unam.mx/ selecciona un día en el calendario.

– Llama a la página http://132.248.6.147/geo/ris.php, recupera con `_GET` la fecha y realiza una conexión ajax.

– Realiza una conexión a la base de datos para obtener: id_muestreo hora_inicio hora_fin ruta_fits

– Se crea un archivo temporal llamado `askitmp $fecha.tmp`.

– `CrearTxt.sh`: Este programa se ejecuta cada 0.1 segundos buscando el archivo `askitmp $fecha.tmp` y ejecuta a su vez el archivo `crearTxt.py` pasándole como parámetro la fecha, borra el archivo temporal `askitmp $fecha.tmp`.

– `CrearTxt.py`: Este programa usa el parámetro de la fecha calcula la ruta donde se encuentra almacenado el archivo fits que contiene los tres canales del telescopio a partir de este, genera un archivo temporal por cada

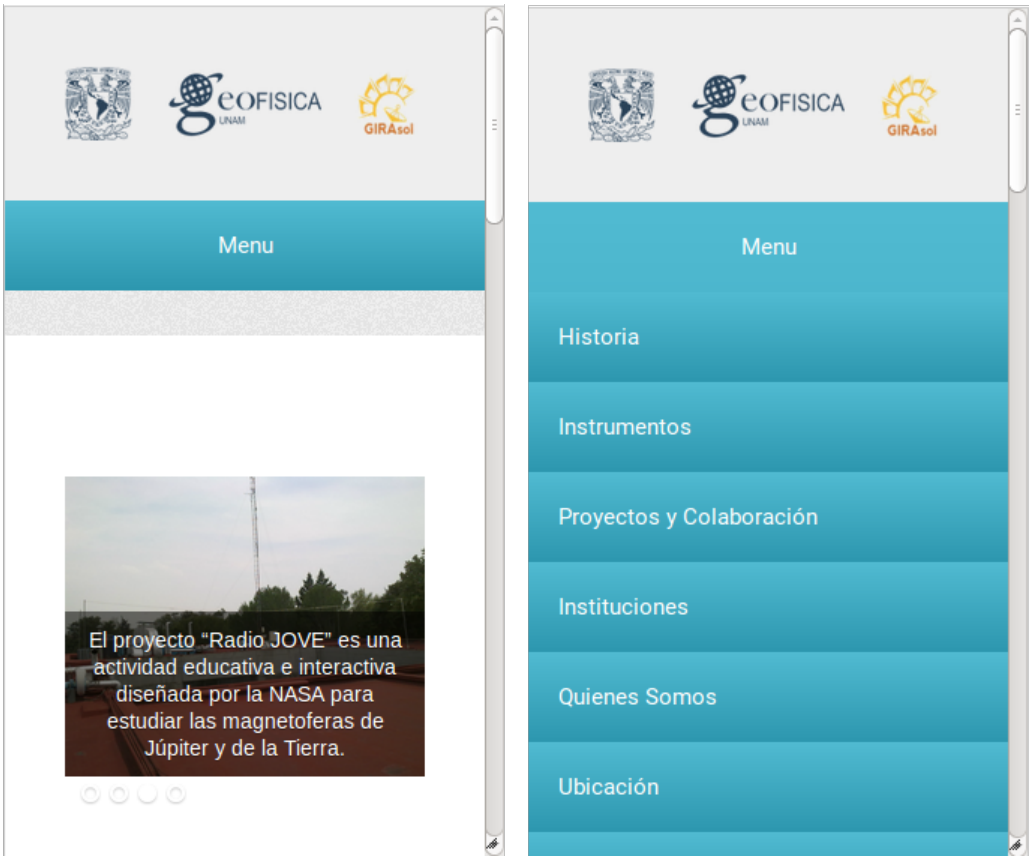


Figura 6.5: Página web en modo responsivo

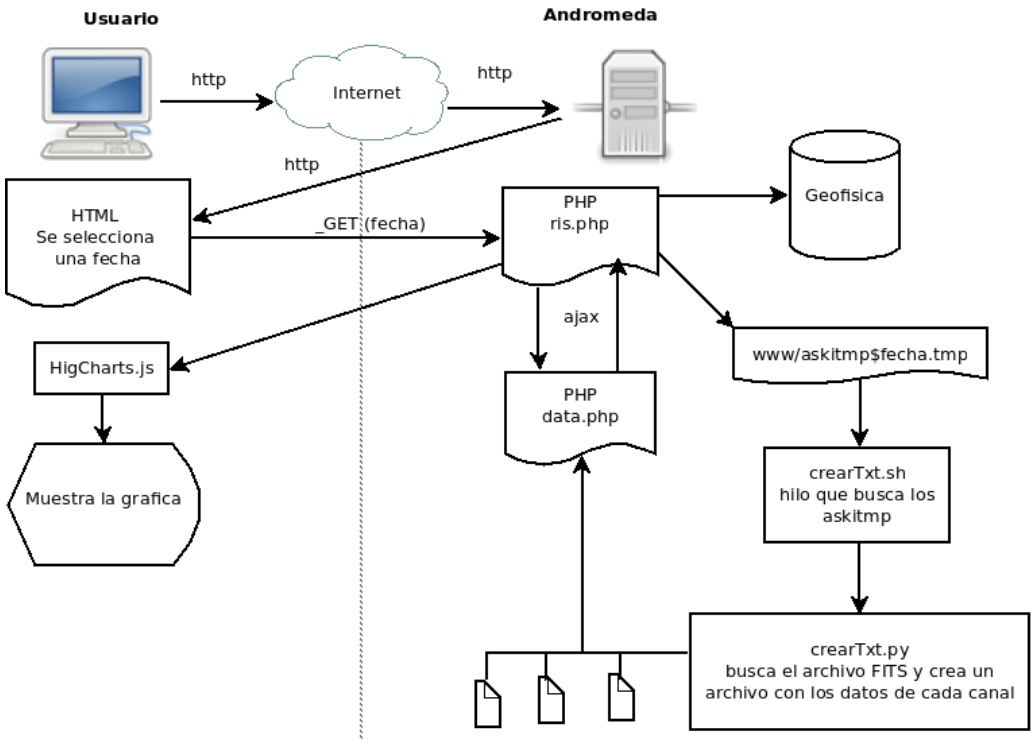


Figura 6.6: Diagrama del algoritmo de Visualización para RIS de los datos históricos

canal. `/risFile/riscanal1 $fecha.txt`, `/risFile/riscanal2 $fecha.txt`, `/risFile/riscanal3 $fecha.txt`.

– Por medio de `ajax` en el archivo `data.phpse` buscan los tres archivos y se guardan en un arreglo que se regresa como parámetro al flujo de la página `index.php`.

– Con la biblioteca HighCharts se reciben los datos y se crea la gráfica.

– El botón descarga recibe la ruta del archivo fits de la consulta de la BD.

Durante el proceso de visualización los datos pasan por tres estados:

1. Los datos están en un archivo FITS que se generó a partir de los datos del telescopio RIS.

2. El archivo FITS es procesado para separar los datos de cada canal.

3. Los datos de cada canal son graficados y mostrados al usuario final.

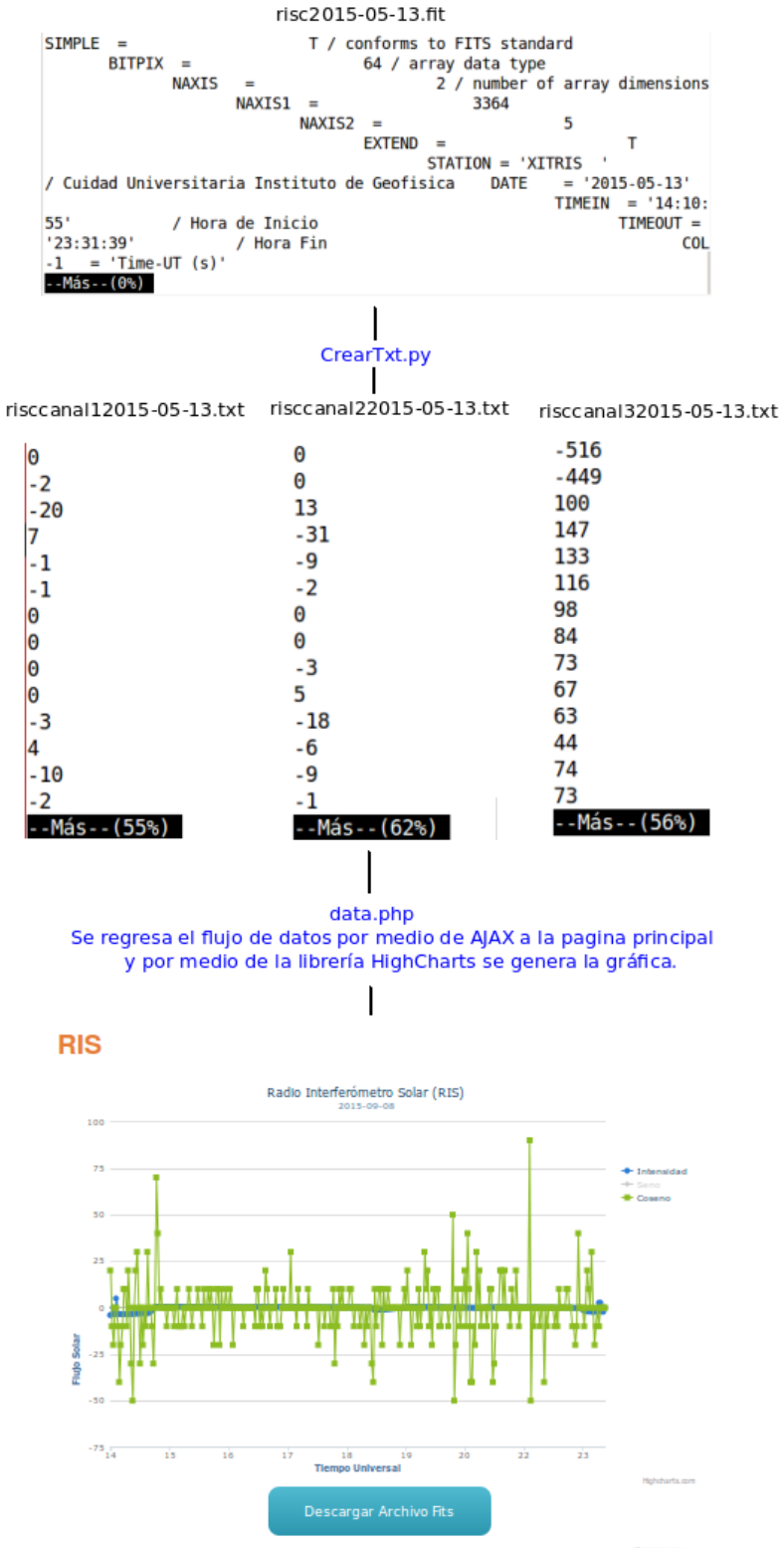


Figura 6.7: Transformación de los datos del RIS

6.3. Diseño del algoritmo de Visualización para RIS en Tiempo Real

El servidor Quetzal, cuenta con una tarjeta de adquisición de datos del telescopio RIS, el software de adquisición de datos genera cada 5 minutos una imagen de la gráfica que muestra los datos de los 4 canales de ese día llamada flujo.png y es colocada en `/home` servidor Quetzal.

Se creo un algoritmo que fue programado en shell llamado `crearImgRisTR.sh`, se encarga de crear una conexión a Quetzal y copia la imagen al servidor Andrómeda en la siguiente ruta: `/var/www/graf/source/tiempoReal/RisTR.png`, solo se ejecuta una sola vez al encender el servidor ya que realiza la conexión y se duerme cinco minutos y vuelve a realizar la copia en un bucle durante todo el día, todo los días.

El programa `crearImgRisTR.sh` se ejecuta desde la carpeta de trabajo de apache: `/var/www/graf/source/tiempoReal/crearImgRisTR.sh`.

Esta imagen renombrada como `RisTR.png`, se manda a llamar desde la página principal con una función llamada `actualizaRis` programada con jQuery para estarla refrescando cada 5 minutos, con el fin de que siempre se muestre la ultima imagen aunque no se éste recargando la página manualmente.

6.4. Diseño del algoritmo de Visualización de datos históricos para CALLISTO

Se realizaron varios diseños antes de llegar al definitivo. Vale la pena exponer y discutir cada uno de los diseños, dado que se adquirió experiencia en cada uno de ellos.

1er Diseño:

A continuación se muestra el algoritmo para el primer diseño:

- Entra a la página http://cintli.geofisica.unam.mx/ selecciona un día en el calendario.
- Crea un archivo `callisto$fecha.tmp`.

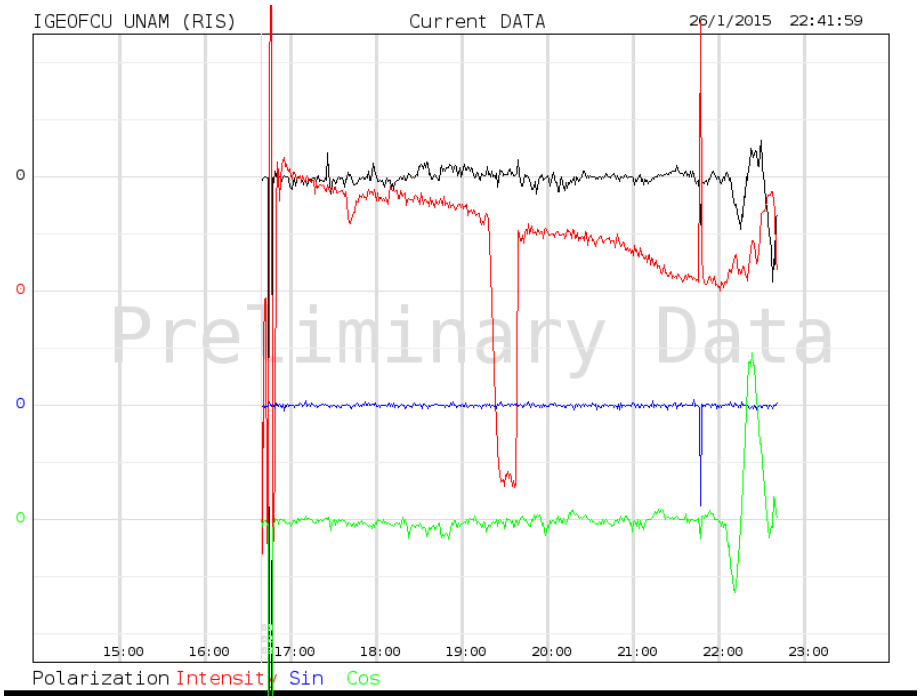


Figura 6.8: Gráfica del RIS en Tiempo Real

– Llama a la página http://132.248.6.147/geo/callisto.php y recupera con `_GET` la fecha, realiza una conexión ajax.

– Se ejecuta `crearImgCallisto.sh` cada 0.1 segundos buscando el archivo `callisto$fecha.tmp` y manda a llamar el programa `crearImg.py`, se le pasa como parámetro la fecha.

– `CrearImg.py`: con el parámetro de la fecha se hace una consulta a la base de datos que regresa las rutas de los archivos fits, se listan y por cada uno se ejecuta `aplpyFITSFigure`, con esto se crea una imagen que se guarda en una carpeta que tiene como identificador la fecha.

– Por medio de `ajax` en el archivo `ajaxCallisto.php` se genera un archivo `zip`, que contendrá los archivos fits dejandolos listos para su descarga.

– Se crea una cuadricula de imágenes que apunta a la carpeta generada.

Conclusión del primer diseño: Cuando se implementó este algoritmo la página tardaba mucho tiempo en cargar ya que el usuario al seleccionar el día, comenzaba la generación de 37 imágenes (que son en promedio el número de imágenes que se generan al día) y se mostraban una vez que todas habían sido generadas y guardadas

en una carpeta temporal en el servidor.

Cada imagen en promedio tarda 30 segundos en ser generada por Python, así que estamos hablando en promedio de 18.5 minutos en cargar la página. El tiempo en que se genera la imagen no se puede optimizar por lo que se tiene que realizar un cambio en la manera de visualizar las imágenes por lo tanto se realizó un nuevo diseño.

2do Diseño:

Las acciones que se ejecutan en el segundo diseño del algoritmo son:

- Entra a la página http://cintli.geofisica.unam.mx/ selecciona un día en el calendario.
- Crea un archivo `callisto$fecha.tmp`.
- Llama a la página http://132.248.6.147/geo/callisto.php y recupera con el metodo `_GET` de PHP la fecha, realiza una conexión ajax.
- Se ejecuta `crearImgCallisto.sh` con una periodicidad de 0.1 segundos buscando el archivo `callisto$fecha.tmp` y manda a llamar el programa `crarImg.py`, se le pasa como parámetro la fecha.
- `CrearImg.py`: usando el parámetro de la fecha se hace una consulta a la base de datos que regresa las rutas de los archivos fits, se listan y por cada uno se ejecuta `aplpyFITSFigure` con esto se crea una imagen que se guarda en una carpeta temporal que tiene como identificador la fecha.
- Se listan los archivos por fecha de la carpeta temporal, cuando encuentra uno se pregunta si ya se generó el siguiente y si es el caso se muestra en la página, si no se espera hasta que este sea creado.
- Este proceso termina al listar los 37 archivos y al finalizar el proceso todos los archivos son cargados en la página.
- Por medio de `ajax` en `ajaxCallisto.php` se genera un archivo zip, que contendrá los archivos fits dejándolos listos para su descarga.
- Se crea una cuadricula de imágenes que apunta a la carpeta generada.

Conclusión del segundo diseño: Este algoritmo a diferencia del primero, no espera a que todas las imágenes sean creadas para cargar la página, si no que cada que una imagen es generada se muestra en la página mientras aparece un cuadro dónde indica que se esta generando la siguiente. Con esto el usuario puede ir analizando los datos de la primer imagen que tardó en cargar aproximadamente 30 segundos

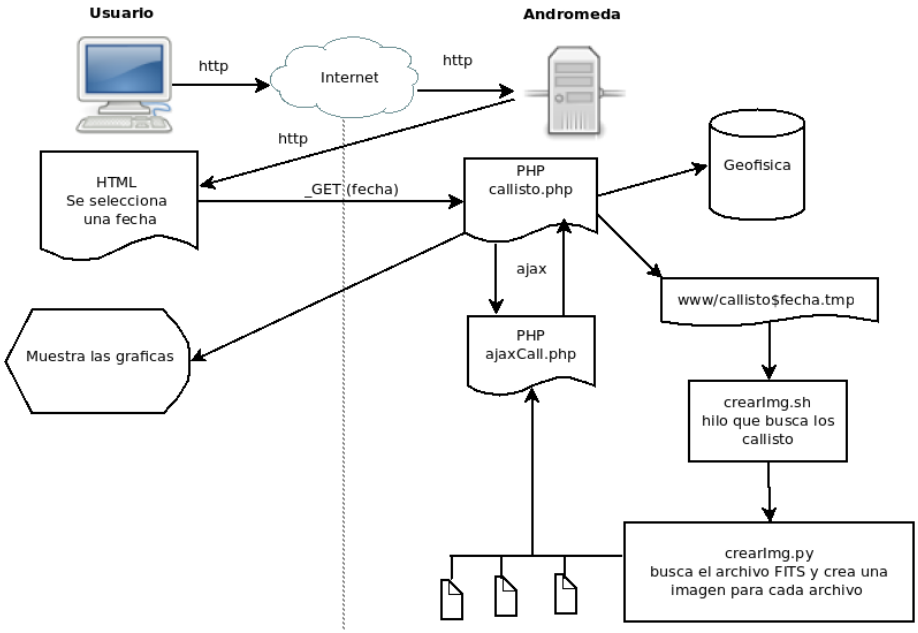


Figura 6.9: Diagrama del algoritmo de Visualización para CALLISTO de los datos históricos 2do diseño

mientras las demás se van generando. Con esto se consiguió no perder el interés del usuario y pensando en que es información de una página de estudio, el usuario esperaría la carga de estas imágenes.

Paralelamente a la implementación de este algoritmo, se desarrollo en Python un filtro para quitar el ruido que tenia que implementarse en la página.

Este filtro sirve para bajar el ruido que existe en los espectros FITS, una vez que se tiene el espectro se obtiene el kernel o núcleo para después normalizar el archivo FITS por completo. Este filtro no elimina el ruido al cien por ciento ya que parte de la señal que existe dentro del ruido es información.

Al realizar este filtrado el servidor tardaba 1 minuto aproximadamente en generar cada imagen, por lo que se tuvo de nuevo el problema del tiempo de carga de la página ya que esperar 37 minutos en su carga total es demasiado tiempo, tiempo que es forzoso esperar ya que son procesos necesarios para generar las imágenes. Por lo que se tuvo que generar un tercer algoritmo de visualización.

3er Diseño

A continuación se muestran los pasos del tercer diseño del algoritmo de

visualización:

- Entra a la página http://cintli.geofisica.unam.mx/ selecciona un día en el calendario.
- Crea un archivo callisto$fecha.tmp.
- Llama a la página http://132.248.6.147/geo/callisto.php y recupera con el metodo `_GET` de PHP la fecha, realiza una conexión `ajax`.
- Se crea una cuadrícula de imágenes que tienen como identificador la ruta de su archivo FITS.
- Cuando se selecciona una imagen se crea un archivo llamado callisto$fecha.tmp que contiene la fecha como identificador único para cada imagen.
- Se ejecuta `crearImgCallisto.sh` con una periodicidad de 0.1 segundos buscando el archivo `callisto$fecha.tmp` y manda a llamar el programa `crearImg.py`, se le pasa como parámetro la fecha.
- `CrearImg.py`: con el parámetro de la fecha se hace una consulta a la base de datos que regresa las rutas del archivo FITS, y ejecuta el filtro para disminuir el ruido.
- Se ejecuta `aplpyFITSFigure` con esto se crea una imagen que es mostrada al finalizar su generación.
- Por medio de `ajax` en el archivo `ajaxCallisto.php` se genera un archivo `zip`, que contendrá los archivos FITS dejándolos listos para su descarga.

Conclusión del tercer diseño: Este diseño a diferencia de los dos anteriores cuenta con la independencia entre las imágenes, se muestra una cuadricula con 37 ligas mostrando la hora y fecha en que fue generado el muestreo una vez que el usuario da clic a la liga comienza la generación de la imagen filtrada.

Este último diseño tiene la ventaja sobre los anteriores de generar sólo las imágenes que son de interés para el usuario ya que los algoritmos anteriores generaban todas las imágenes aunque el usuario sólo quisiera examinar algunas de ellas.

Con esto el tiempo de carga se reduce y el trabajo de procesamiento del servidor también se reduce considerablemente ya que no tiene que estar generando las 37 imágenes forzosamente en cada carga.

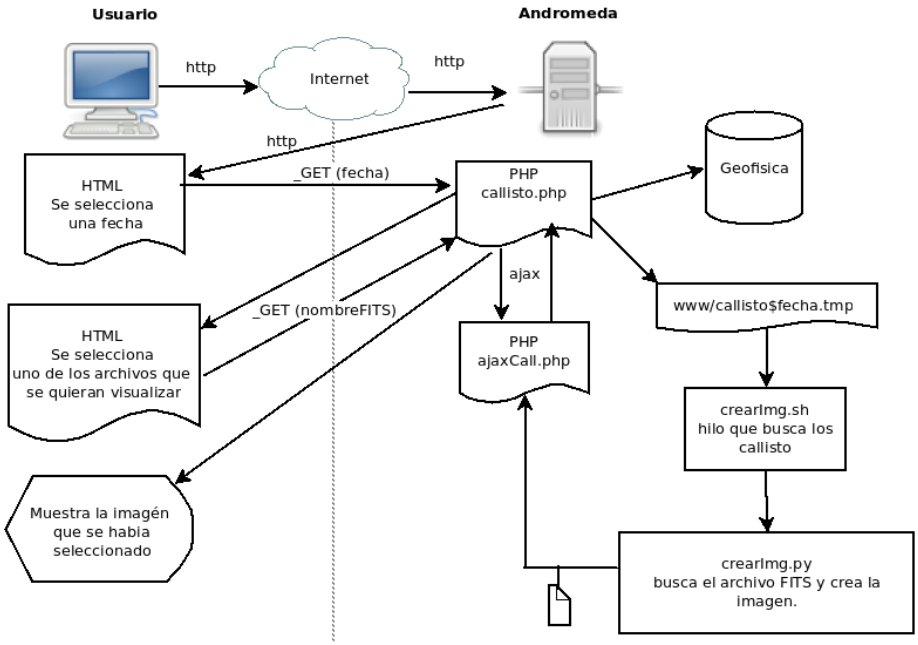


Figura 6.10: Diagrama del algoritmo de Visualización para CALLISTO de los datos históricos 3er diseño

Durante el proceso de visualización los datos pasan por tres estados:

1. Los datos están en un archivo FITS que genera el telescopio CALLISTO.

2. El archivo FITS es procesado para crear una imagen.

3. La imagen creada es filtrada para quitar el ruido.

La siguiente imagen 6.11 muestra la transformación de los datos al pasarla por estos tres estados.

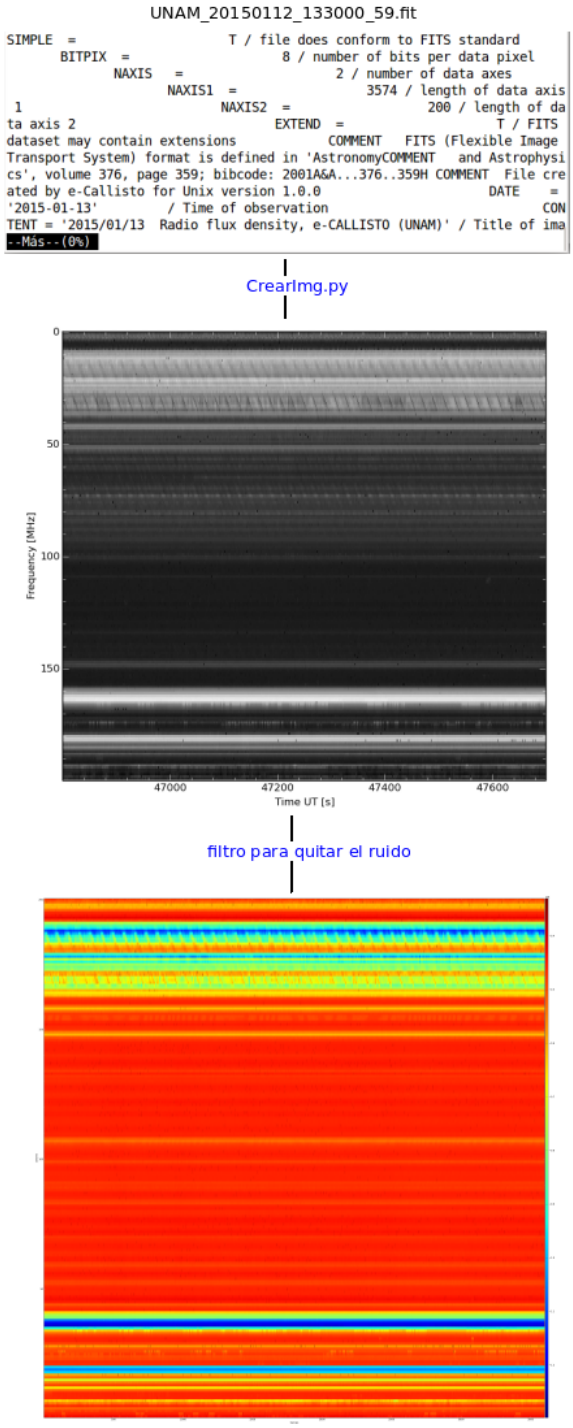


Figura 6.11: Transformación de los datos de CALLISTO

6.5. Diseño del algoritmo de Visualización para CALLISTO en Tiempo Real

El telescopio CALLISTO genera cada 15 minutos un archivo FITS y es guardado en el servidor CALLISTO.

El programa `crearImgCallistoTR.sh` cada 15 minutos crea una conexión de Andrómeda a CALLISTO busca el último archivo que se generó, con el comando `scp` crea una copia del archivo FITS en `/var/www/graf/source/tiempoReal/ CallistoTR.fit`, al terminar la copia del archivo se manda a ejecutar el programa `crearImgTR.py`.

`crearImgTR.py`, es un programa en Python que se encarga de procesar el archivo FITS y generar una imagen llamada `CallistoTR.png`, esta imagen se estará reemplazando por la ultima cada 15 minutos.

Esta imagen se manda a llamar desde la página principal con una función llamada actualiza programada con jQuery para estarla refrescando cada 15 minutos, con el fin de que siempre se muestre la última imagen aunque no se este recargando la página manualmente.

Figura 6.12: Gráfica del CALLISTO en Tiempo Real

Capítulo 7

Configuración del servidor.

Este capítulo se divide en tres secciones, en la primera sección se ilustra la instalación de Debian 6, se eligió este sistema operativo porque es el que se utiliza en el laboratorio, durante la instalación se configurará la tarjeta de red para que funcione como servidor web.

En la segunda sección se trata la configuración del sistema operativo, se instalarán las bibliotecas necesarias de java, Python y C, también se instalarán y configurarán los servidores Mysql Apache y PHP para que los programas que componen el Sistema Andrómeda se ejecuten correctamente.

En la última sección se mostrará como inicializar los programas que componen el Sistema Andrómeda.

Es muy importante documentar estos pasos dado que el laboratorio no cuenta con personal especializado y en caso de falla o mantenimiento cualquier persona debe ser capaz de solucionar un problema usando como base el presente trabajo.

7.1. Instalación del sistema operativo

Sistema Operativo Debian El Sistema de Andrómeda esta corriendo sobre un sistema Debian versión 6, para instalarlo se debe descargar de preferencia la última versión del portal https://www.debian.org/distrib/netinst eligiendo la versión para la arquitectura de 64 bits. Se realizará una instalación estándar, sólo teniendo en cuenta los siguientes datos para configurar la red:

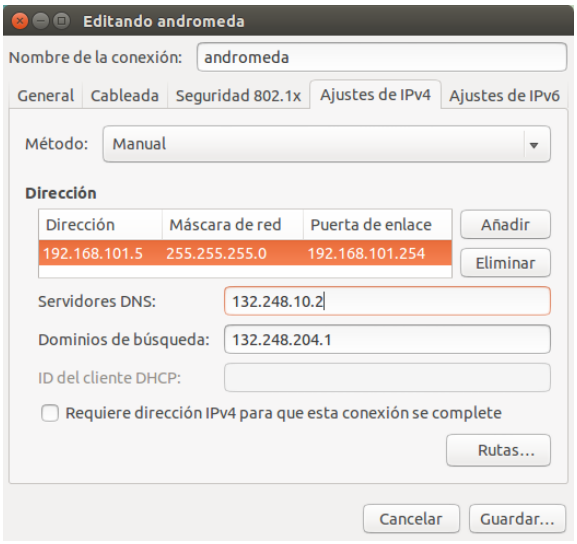


Figura 7.1: Datos para configurar la Red

Cuando la instalación haya concluido deberá actualizar el sistema como root con los siguientes comandos:

```
apt-get update && apt-get upgrade
```

7.2. Ambientación del Sistema Operativo:

Una vez que el sistema operativo esté listo, es necesario instalar bibliotecas y programas para que el sistema pueda funcionar correctamente:

- C

El programa xdat2txt está elaborado en lenguaje C, por lo que se debe instalar gcc ejecutando como root los siguientes comandos.

```
apt-get install gcc-c++
apt-get install gcc
apt-get install g++
```

- JAVA

Los programas `almacenaCallisto.jar` y `almacenaRis.jar` fueron desarrollados en java, los programas ya están compilados y listos para su ejecución por lo que solo será necesario instalar el JRE (Java Runtime Environment) que es la máquina virtual de java para ejecutar programas en este lenguaje. Si se requiere

de algún cambio en el código fuente del programa es necesario instalar el JDK que incluye el compilador de java para poder compilar los cambios. La instalación viene descrita paso a paso en la pagina oficial de java:

JAVA:

http://www.oracle.com/technetwork/es/java/javase/downloads/index.html

JRE (Java Runtime Environment):

http://docs.oracle.com/javase/8/docs/technotes/guides/install/linux_jre.html#CFHBJIIG

JDK (Java Development Kit):

http://docs.oracle.com/javase/8/docs/technotes/guides/install/linux_jdk.html#A1098871

- SSH

Como ya se había explicado en el capítulo 3 se utiliza ssh para el intercambio de archivos por lo que es necesario ejecutar el siguiente comando:

```
apt-get install openssh-3.5p1-6
```

La autenticación se realiza mediante el algoritmo de cifrado de llaves públicas RSA. Este algoritmo de cifrado es todo un paradigma de seguridad en redes que en este trabajo solo lo utilizaremos sin entrar a fondo en su funcionamiento.

SSH cuenta con un programa llamado ssh-keygen que genera un par de llaves, una pública con extensión .pub y una privada sin extensión, que son guardadas en la carpeta /etc/ssh.

Se ejecuta el comando y luego se entra a la carpeta para verificar que las llaves existan:

```
ssh-keygen
cd /etc/ssh
```

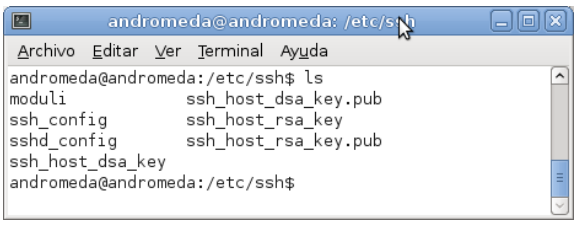


Figura 7.2: Llaves generadas para la autenticación

Este algoritmo funciona mediante el intercambio de llaves por lo que se

deberá mandar la llave publica al servidor y en Andrómeda se quedará la privada que es la que no tiene extensión, este par de llaves servirá para autenticarnos con los servidores que queremos tener comunicación por medio de scp, en este caso es RIS y CALLISTO ejecutando los siguientes comandos en una terminal de Andromeda:

RIS:

```
scp id_rsa.pub usuario_servidor@direccion_servidor.com:/tmp
```

CALLISTO:

```
scp id_rsa.pub usuario_servidor@direccion_servidor.com:/tmp
```

Se pedirá la contraseña del servidor con esto estaremos conectados remotamente al servidor. Entramos a la carpeta temporal /tmp y añadimos la llave a las llaves autorizadas con el siguiente comando:

RIS:

```
cd /tmp
cat id_dsa.pub /.ssh/authorized_keys
```

CALLISTO:

```
cd /tmp
cat id_dsa.pub /.ssh/authorized_keys
```

por ultimo borramos las llaves de la carpetas temporales

RIS:

```
rm id_dsa.pub
```

CALLISTO:

```
rm id_dsa.pub
```

Con esto se podrá tener acceso con la autenticación a través de llaves públicas por lo que no se deberá introducir la contraseña cada vez que se realice una conexión SSH. Se recomienda realizar todo el proceso primero para un servidor y después para el otro.

- PYTHON

Python viene ya incorporado en Debian por lo que no es necesario instarlo, sin embargo hay que instalar una serie de bibliotecas para el funcionamiento de los programas que fueron realizados con este lenguaje.

Estas bibliotecas deben ser descargadas de sus sitios y descomprimidas en la carpeta /etc de Andrómeda una vez que se encuentren ahí, hay que entrar a estas

carpetas y tan solo se deberá ejecutar el comando:

```
python setup.py install
```

Se debe respetar la secuencia ya que existen dependencias entre ellas.

- Astropy: http://www.astropy.org/

- Matplotlib: http://matplotlib.org/

- Pyfits: http://www.stsci.edu/institute/software_hardware/pyfits/release

- MySQLdb: https://pypi.python.org/pypi/MySQL-python/1.2.5

Hasta este momento se ha configurado la parte del manejo de archivos FITS, hace falta realizar la parte de almacenamiento y visualización, para ello instalaremos Apache, MySQL y PHP en ese orden:

- APACHE

Para instalar apache es necesario ejecutar el siguiente comando:

```
apt-get install apache2
```

Con este comando Apache debe quedar funcionando, para verificar se ejecutará el comando:

```
service apache2 status
```

y su salida será, la que se muestra en la figura 7.3

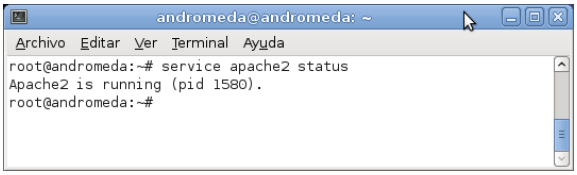


Figura 7.3: Salida de consola para el comando service apache2 status

Su directorio de trabajo por defecto es: `/www/var` en esta carpeta es donde se aloja la pagina web con los archivos necesarios para su correcto funcionamiento.

- MYSQL

El manejador de base de datos MySQL se instala de la siguiente manera:

```
sudo apt-get install mysql-server mysql-client
```

Al momento en que se realiza la instalación se va a solicitar la contraseña para el usuario root, por lo que se debe proporcionar. Cabe mencionar que no es el usuario root de Linux y la contraseña puede ser la que ya tiene el servidor Andrómeda o se puede cambiar, las demás configuraciones se dejan con los valores por defecto.

Para el Sistema Andrómeda se deben crear dos bases de datos:

geofísica: Aquí se guardan todos los datos de RIS y CALLISTO.

web: Esta BD sera la que utilice Wordpress para la página web.

Para crearlas se debe abrir una terminal en Andrómeda y escribir el siguiente comando:

```
mysql -u root -p
```

Este comando pedirá la contraseña que se asignó a root en el momento de la instalación, una vez que se escribe liberará el prompt de MySQL, donde se crearán las bases de datos escribiendo:

```
create database geofisica;
create database web;
```

Para comprobar que todo quedó correctamente instalado se ejecutará: **use geofisica** y **use web**, con esto entramos a la base de datos que ha sido creada y deberán salir mensajes similares a los que se muestra en la figura 7.4 para cada una de las bases de datos.

```
mysql> use geofisica
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>
```

Figura 7.4: Salida de consola para el comando use geofisica

Con esto ya tenemos listo MySQL para continuar la configuración.

- PHP

Para la instalación de PHP es necesario ejecutar:

```
apt-get install php5 php-pear php5-mysql
```

Para asegurar su correcto funcionamiento se crea un archivo en el directorio de apache llamado info.php de la siguiente manera:

```
nano /var/www/info.php
```

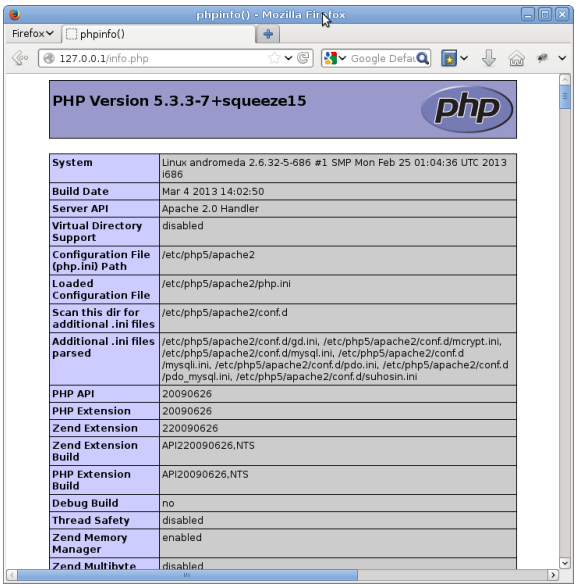
y se escribirá lo siguiente:

```
<?php phpinfo(); ?>
```

Guardamos el archivo y reiniciamos el servicio apache2 para que se cargue con php:

```
service apache2 restart
```

Una vez hecho esto se debe entrar en un navegador a localhost/info.php para verificar que php este funcionando. Deberá salir una pantalla como la que se muestra en la figura 7.5 .



System	Linux andromeda 2.6.32-5-686 #1 SMP Mon Feb 25 01:04:36 UTC 2013 i686
Build Date	Mar 4 2013 14:02:50
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/gd.ini, /etc/php5/apache2/conf.d/mcrypt.ini, /etc/php5/apache2/conf.d/mysql.ini, /etc/php5/apache2/conf.d/mysqli.ini, /etc/php5/apache2/conf.d/pdo.ini, /etc/php5/apache2/conf.d/pdo_mysql.ini, /etc/php5/apache2/conf.d/suhosin.ini
PHP API	20090626
PHP Extension	20090626
Zend Extension	220090626
Zend Extension Build	API220090626,NTS
PHP Extension Build	API20090626,NTS
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
Zend Multibyte	disabled

Figura 7.5: Información de la configuración por defecto de PHP

- WORDPRESS

Este es un manejador de contenido muy fácil de instalar, hay que entrar a su página y descargar la última versión.

http://es.wordpress.org/

Se debe descomprimir el archivo dentro de la carpeta /www/var

Abre el archivo wp-config-sample.php con un editor de texto plano y se llenan los datos de la conexión a la base de datos.

define('DB_NAME', 'web');

define('DB_USER', 'root');

define('DB_PASSWORD', 'passwordRootMySQL');

define('DB_HOST', 'localhost');

Se guarda el archivo como **wp-config.php.** En un navegador se entra a **/localhost/wp-admin/install.php** y se llena el formulario que se muestra.

Con esto queda listo wordpress para albergar la pagina web.

7.3. Configuración el Sistema Andrómeda:

Hasta este momento se ha instalado y configurado Debian para que todo funcione correctamente, sin embargo falta configurar el Sistema Andrómeda.

Se hizo entrega de un dispositivo de almacenamiento con tres carpetas; cada carpeta se refiere a una parte que compone el Sistema Andrómeda.

Recepción y almacenamiento de Datos:

-recepcionRis.jar

-recibeRis.sh

-recepcionCallisto.jar

-recepcionArchivos.sh

-crearImgCallisto.sh

-callisto.conf

-ris.conf

Almacenamiento histórico de datos:

-baseGeofisica.sql

Visualización:

-/www

-webGeofisica.sql

Estos archivos se tienen que colocar en la carpeta de trabajo del Sistema Andrómeda que es `/media/database`. En esta ruta se encuentran dos carpetas; *RIS* y *CALLISTO*, que guardan los archivos FITS.

Si por algún motivo la configuración del servidor cambia, estas carpetas deben ser creadas y se deben colocar los archivos FITS de los telescopios, quedando la configuración que se muestra en la figura 7.6.

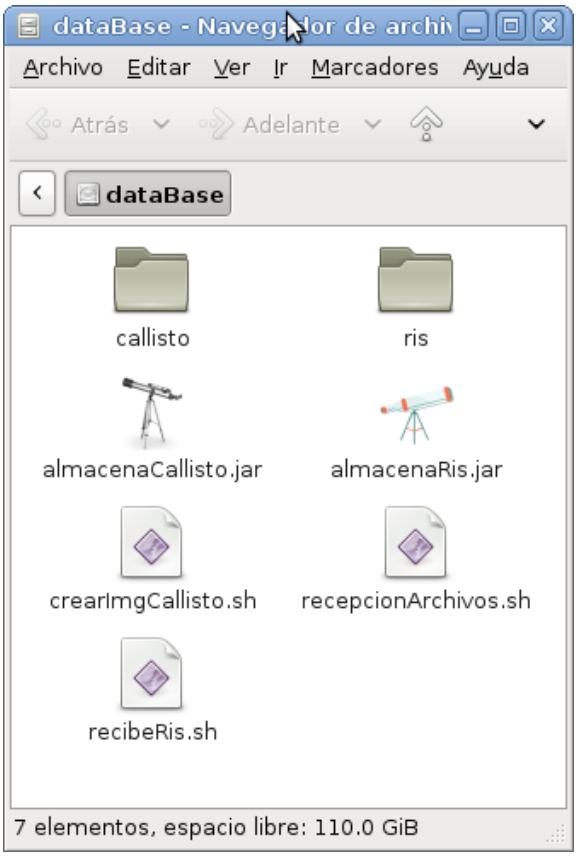


Figura 7.6: Configuración de la carpeta dataBase

Los archivos `recepcionArchivos.sh` y `recibeRis.sh` son los scripts que se encargan de ejecutar periódicamente los archivos jar y estos a su vez, se encargan de hacer el proceso de recepción y carga de datos diarios en la base de datos *Geofísica*.

El archivo `crearImgCallisto.sh`se mandará ejecutar por medio del script `recepcionArchivos.sh`, por lo que no se deberá ejecutar en consola.

La manera de ejecutar los archivos es muy sencilla, solo se deben escribir las lineas que se muestran en la figura 7.7.

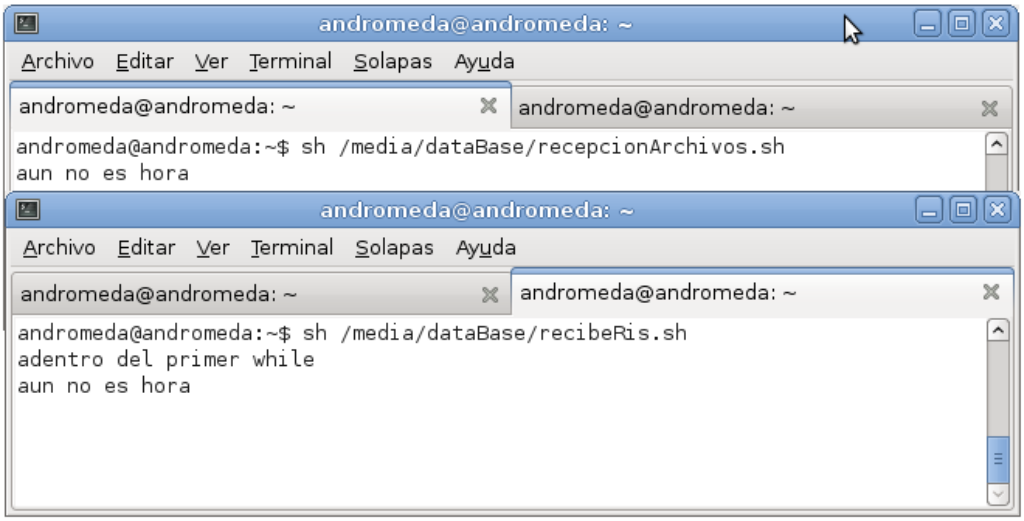


Figura 7.7: Configuración de la carpeta dataBase

Los archivos `ris.conf` y `callisto.conf` son de configuración y deben copiarse en `/etc` , tendrán escrita la contraseña y usuario de MySQL, también la IP del servidor de la base de datos.

Bajo este escenario tiene como `URL: 127.0.0.1`, en el caso de que el servidor MySQL cambiará de IP, solo se tiene que cambiar esta línea por la nueva IP. Estos cambios se deben realizar en los dos archivos.

También existe una variable llamada ruta, sirve para cambiar el lugar de almacenamiento de los archivos, en este caso es `/media/database`, si se cambiaran de lugar, la ruta debe cambiarse y los archivos `sh`se deben ejecutar en el nuevo espacio de trabajo.

Almacenamiento histórico de datos:

Para el almacenamiento histórico de datos se tienen que hacer dos cosas:

1. Hacer la carga en el servidor MySQL
2. Copiar las carpetas donde se encuentran los archivos FITS al espacio de trabajo.

Para este proceso se deja el archivo SQL `baseGeofisica.sql`, los archivos FITS están en el servidor Andrómeda en las carpetas `/media/database/ris` y `/media/database/callisto`.

Por medio de phpMyAdmin es muy fácil cargar la base de datos, a continuación

se muestran los pasos a seguir:

Desde un navegador entrar a la dirección: http://127.0.0.1/phpmyadmin/ como se muestra en la figura 7.8, donde se debe introducir el usuario y la contraseña.

Figura 7.8: Iniciar sesión desde phpMyAdmin

Después dar clic en el botón de importar ver figura 7.9:

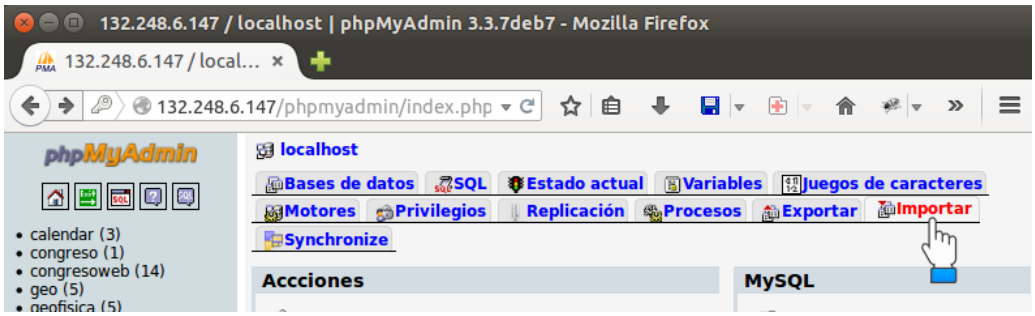


Figura 7.9: Configuración de la carpeta dataBase

Se mostrará una ventana como en la figura 7.10, dar clic en el botón examinar, y se busca el archivo `baseGeofisica.sql`.

Figura 7.10: Configuración de la carpeta dataBase

Por último dar clic en continuar como se muestra en la figura 7.11:

Figura 7.11: Configuración de la carpeta dataBase

Visualización:

Para la visualización existen dos partes:

1. Hacer la carga en el servidor MySQL de la base de datos que tiene la configuración de wordpres:

2. Copiar la carpeta `www` al espacio de trabajo de apache y php.

Para cargar la base de datos con el archivo `webGeofisica.sql` se siguen los mismos pasos que cuando se cargo `baseGeofisica.sql`.

La segunda parte es tan simple como copiar toda la carpeta *www* que se encuentra en los archivos entregados y pegarla en el espacio de trabajo de apache, que en la mayoría de los casos es `/var/www`.

Para revisar que todo haya salido bien entrar a la dirección http://cintli.geofisica.unam.mx/ desde una red externa, y se debe de visualizar el portal, como se muestre en la figura 7.12.

Figura 7.12: Configuración de la carpeta dataBase

Capítulo 8

Conclusiones

El Sistema Andrómeda es capaz de realizar la transformación y el almacenamiento de los datos de los telescopios RIS y CALLISTO diariamente. El portal web está en funcionamiento, se pueden consultar datos generales del laboratorio del Radio Observatorio Solar del Instituto de Geofísica de la UNAM, también se pueden consultar los datos históricos de los archivos que han generado los telescopios RIS y CALLISTO, sus archivos FITS están disponibles para cualquier persona que le interese descargarlos para su estudio.

Durante el desarrollo del Sistema Andrómeda el primer problema a resolver fue la unificación de los archivos al formato FITS, se realizaron pruebas con diferentes bibliotecas aprobadas por la comunidad de la NASA, teniendo éxito con la biblioteca PyFITS en lenguaje python, el archivo generado fue correctamente leído por los programas que se utilizan en el laboratorio y verificado correctamente por el portal de la NASA. Por lo que este problema se resolvió satisfactoriamente.

Se automatizó la copia diaria de los archivos tanto de RIS como de CALLISTO al servidor Andrómeda con el protocolo SSH y se optó por una autenticación de llaves RSA. Esta parte se programó con scripts shell ya que solo se utilizan comandos del sistema. Con esto se asegura que la comunicación y transferencia de archivos sea segura y estable.

Una vez que los archivos se copiaron a Andrómeda se realizó el análisis para el diseño de la base de datos, durante este proceso se tomó en cuenta que es una base de datos con consultas fijas, que se incrementa diariamente, RIS recopila un

promedio de 138.20 KB y CALLISTO 760.30 KB, también se tomó en cuenta que la estructura pueda albergar otras tablas para otros instrumentos en un futuro. El DER esta normalizado en su tercera forma normal y las consultas que se realizan tardan 0.0302 segundos.

En el proceso de transformación y carga de datos se utilizó el lenguaje java, para los archivos que llegan de CALLISTO se guardan sus datos generales y la ruta donde se guardará el archivo en la tabla `muestreoCallisto`. Para el caso de RIS llegan los 4 archivos `xdat`, tienen que pasar por un proceso de transformación para crear el archivo FITS, se guardan sus datos y la ruta donde se guardará el archivo en la tabla `muestreoRis`.

Para la carga de los datos históricos se utilizó el mismo algoritmo pero de manera recursiva para guardar todos los datos de los años anteriores, primero se realizó la carga del RIS sin ningún inconveniente, al realizar la carga de CALLISTO el disco del servidor Andrómeda se saturó, y se tuvo que conectar un disco duro externo.

El mayor inconveniente al programar la parte de visualización, fue para los datos de CALLISTO, el archivo FITS debe pasar por dos procesos; el primero es un filtro y el segundo la transformación a una imagen PNG, este proceso se debe repetir 37 veces que es el número de archivos que CALLISTO genera en un sólo día. Se realizaron tres diseños en la presentación de las imágenes y el tiempo de carga se redujo de 20 minutos a 3 minutos. Cabe mencionar que este último algoritmo se probó en una máquina con un procesador core i5 y el proceso tardó 30 segundos.

Sobre el Sistema Andrómeda se pueden desarrollar diversos trabajos de investigación ya que es una base de datos histórica dónde se pueden descubrir patrones, predicciones y se pueden hacer diversas consultas para el estudio de los fenómenos solares a través del tiempo.

Con base a lo anterior podemos asegurar que el objetivo planteado al inicio del trabajo se cumplió completamente.

Como trabajo a futuro, el siguiente paso para complementar el sistema podría ser un algoritmo que al momento de estar guardando los archivos FITS tenga la capacidad de detectar un fenómeno solar, enviar correos con los archivos FITS a personas que se hayan inscrito previamente en un formulario en la página web para recibir estas alertas.

También se podría realizar el diseño para guardar los archivos de la estación Radio JOVE y la estación LAVNET-México.

Otra mejora considerable e inmediata es migrar el sistema a un equipo con mayores recursos, para que sea más eficiente.

Bibliografía

[1] Beynon-Davies, Paulautor, *Sistemas de bases de datos / Paul Beynon-Davies ; versión española coordinada y traducida por Enrique Alegre* Barcelona ; Bogota ; Buenos Aires ; Caracas ; México : Reverté, [2014].

[2] Mora Rioja, Arturo, *Bases de datos :diseño y gestión* Madrid : Editorial Sintesis, 2014.

[3] DuBois, Paul, *MySQL* Upper Saddle River, New Jersey ; México City : Addison-Wesley, c2009.

[4] Victor Hugo de la Luz Rodriguez, *Desarrollo de sofware para la captura, despliegue, almacenamiento, procesamiento y publicación de datos en tiempo real obtenidos del radio interferometro solar ris* Universidad Nacional Autonoma de Mexico. Facultad de Ciencias, institución que otorga el título.

[5] Paz Martínez, Gaudencio, *Desarrollo del espectrómetro Callisto-México* Universidad Nacional Autónoma de México. Facultad de Ingeniería, institución que otorga el título.

[6] Debian, *https://www.debian.org/index.es.html* 2014

[7] Documentación MySQL, *https://www.mysql.com/* 2015

[8] Documentación Apache, *http://www.apache.org/* 2015

[9] Documentación pyfits 3.3, *https://pypi.python.org/pypi/pyfits/3.3* 2015.

[10] SHELL, *http://linuxcommand.org/lc3_wss0010.php* 2015.

[11] Documentación JAVA, *https://www.java.com/es/* 2015.

[12] Documentación JQuery, *https://jquery.com/* 2015.

[13] Documentación FITS, *http://fits.gsfc.nasa.gov/* 2015.

Índice de figuras