UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES

DESARROLLO DE UN SISTEMA DE ACCESO A HISTORIAS MÉDICAS DE UNA CLÍNICA A TRAVÉS DE DISPOSITIVOS MÓVILES.

TRABAJO ESPECIAL DE GRADO

Presentado ante la

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

Como parte de los requisitos para optar al título de

INGENIERO EN TELECOMUNICACIONES

REALIZADO POR Jesús D. Pacheco C.

Javier I. Val de los R.

PROFESOR GUÍA Lic. Iván Escalona

FECHA Caracas, 24 de septiembre de 2012

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES

DESARROLLO DE UN SISTEMA DE ACCESO A HISTORIAS MÉDICAS DE UNA CLÍNICA A TRAVÉS DE DISPOSITIVOS MÓVILES.

REALIZADO POR Jesús D. Pacheco C.

Javier I. Val de los R.

PROFESOR GUÍA Lic. Iván Escalona

FECHA Caracas, 24 de septiembre de 2012

DESARROLLO DE UN SISTEMA DE ACCESO A HISTORIAS MÉDICAS DE UNA CLÍNICA A TRAVÉS DE DISPOSITIVOS MÓVILES.

Jesús D. Pacheco C.
jesuspacheco76@gmail.com

Javier I. Val de los R.
valjavierster@gmail.com

Resumen

La creencia de que la tecnología solo hace encarecer los servicios ya se ha convertido en un mito, el mundo vive una etapa en la que las comunicaciones juegan un papel importante en la vida de sus habitantes, desde la necesidad de mantenernos informados en tiempo real hasta la indispensable comunicación con nuestro entorno.

Confiando en que todos los problemas de la sociedad venezolana tienen reparo, y aportando todo nuestro conocimiento, se plantea el siguiente Trabajo Especial de Grado, el cual busca ayudar al sector salud, y demostrar que la tecnología, con un costo bajo y de implementación sencilla, puede solucionar problemas importantes en nuestra sociedad.

Entendiendo que los tiempos de espera en salas de consulta y emergencia son vitales, este Trabajo Especial de Grado busca dar una solución fácil, económica y segura, que permita al médico tratante obtener de manera inmediata la historia médica del paciente almacenada en un servidor FTP a través de redes WiFi y 3G.

Apoyándonos en tecnologías de identificación, sistemas operativos de alto rendimiento para dispositivos móviles, las redes móviles disponibles y protocolos de trasferencia de archivos se logró darle solución a este problema. Esto es resultado de una investigación teórica y pruebas del sistema, comprobando así el funcionamiento de éste.

A continuación se plantea el estudio teórico, metodológico y práctico de este Trabajo Especial de Grado.

Palabras clave: WiFi, 3G, sencillo, transferencia, archivos.

Dedicatoria

A mis padres por su quisquillosa preocupación, a Juanfer por todos los partidos que no jugamos, a Daniela por todas las películas que no vimos y a Jesús por todos los hoyos que no jugamos.

Javier I. Val D

A mi familia, a todos los que me rodean, y ayudan a seguir día a día, a todos los que me apoyan y están siempre detrás de mí.

Finalmente, a ti, que siempre soñaste con este día.

"Todo lo que soy, o espero ser, se lo debo a mi madre"

Abraham Lincoln

Jesús D. Pacheco C.

Agradecimientos

Agradecemos a todas las personas que hicieron posible este trabajo, nuestras familias y amigos que estuvieron siempre a nuestro lado.

A aquellas personas e instituciones que brindaron su apoyo y creyeron en el proyecto: Clínica Sanatrix, Dr. José San Miguel, Dr. Bernardo Mirabal, Sr. Jean Paul Zerpa, Srta. Fátima Vieira y Sr. John R. Applewhite.

Agradecemos a Daniela y Cristina Márquez, cuya ayuda ha sido de gran importancia para la presentación de este proyecto.

También al ingeniero Ignacio Val por su paciencia y serenidad.

Especial reconocimiento a nuestro tutor Lic. Iván Escalona y a nuestro revisor Ing. Wilfredo Torres, quienes nos apoyaron desde el principio, y cuya guía nos permitió terminar satisfactoriamente nuestro proyecto.

"No solo no hubiéramos sido nada sin ustedes, sino con toda la gente que estuvo a nuestro alrededor desde el comienzo, algunos siguen hasta hoy.

GRACIAS, TOTALES."

Gustavo Cerati

Índice General

Índice de Ilustraciones

Índice de Tablas

Introducción

El interés en nuevos campos de las Telecomunicaciones y la necesidad de apoyar tecnológicamente uno de los sectores más necesitados e importantes de nuestra sociedad, impulsa el desarrollo de este Trabajo Especial de Grado.

La telemedicina es una rama de las telecomunicaciones que se encarga de brindar soluciones de comunicación entre médicos y pacientes. Así como, el rápido y acertado diagnóstico a través del procesamiento de imágenes y resultados de laboratorio. En este caso se busca que las telecomunicaciones y la telemedicina ayuden a establecer una nueva metodología en la forma en que se manejan los archivos médicos dentro de una institución de la salud.

El siguiente Trabajo Especial de Grado contempla el estudio e implementación de un sistema que centralice la información de cada paciente en un servidor con una identificación única para cada paciente, y que esta información pueda ser accedida de manera inmediata a través de una aplicación móvil en un dispositivo con sistema operativo Android.

A continuación se presenta el desarrollo de este Trabajo Especial de Grado en capítulos. Se efectúa el estudio teórico y metodológico, se demuestra el funcionamiento de este sistema, y se ejecuta la puesta en marcha del mismo en un ambiente real, en una Institución Médica de la ciudad de Caracas.

Al final de este trabajo se describen las conclusiones obtenidas del estudio. Así como recomendaciones que ayuden a futuros trabajos a mejorar y ampliar el funcionamiento del sistema.

Capitulo 1. Planteamiento del Proyecto

1.I Planteamiento del Problema

Uno de los problemas que más afecta a los pacientes en las clínicas y hospitales es el tiempo de espera en emergencia y en consulta.

Una de las causas de este retraso, es que en muchas oportunidades las historias médicas de los pacientes no se encuentran a la mano. Éstas se pueden hallar en cuartos llenos de archivos y papeles, que pueden estar lejos de donde están los pacientes. Cuando un doctor solicita una historia, un encargado la busca en estos archivos para luego enviarla al departamento donde se necesita. Es aquí donde el doctor la recibe. Esta búsqueda puede tomar desde pocos minutos, hasta horas. En este último caso, retrasando la atención del paciente.

Con el fin de reducir sustancialmente el tiempo en que las historias llegan a los médicos, se propone:

Un sistema de acceso en tiempo real a historias médicas digitalizadas a través de dispositivos móviles.

La creación de un servidor que contenga todas las historias médicas digitalizadas, lográndose centralizar la información para acceder a ésta de manera remota.

La utilización de las herramientas disponibles (*smartphone* o tableta). Aprovechando las ventajas ofrecidas por los proveedores de servicios móviles, el médico podrá, desde cualquier dispositivo equipado con sistema operativo Android,

acceder a la historia médica de cualquier paciente, en cualquier parte de la clínica, e incluso, si se quiere, desde cualquier parte del mundo.

Gracias a protocolos de trasferencia de datos móviles como UMTS (*Universal Mobile Telecomunication System)* y WiFi (Norma IEEE 802.11x), este sistema tendrá como principal característica la adaptabilidad a cualquiera de estas tecnologías, permitiendo así, que el interesado pueda acceder remotamente a la historia médica de cualquier paciente desde cualquier parte de la clínica, lugar de la ciudad, o desde cualquier parte del país o del mundo.

Para garantizar la seguridad del sistema, y su uso adecuado, cada paciente contará con una identificación, la cual permitirá nombrar y ubicar su historia en el servidor. El doctor leerá esta identificación (con su móvil o tableta). Esta lectura será enviada automáticamente al servidor donde se buscará la historia requerida y luego es enviará de vuelta al dispositivo.

Esta identificación se realizará con códigos QR (*Quick Response Barcodes*). Esta es una tecnología de identificación que, a pesar de no ser costosa ni muy elaborada, ha tenido mucho auge en comunicaciones y en identificaciones móviles dada su versatilidad de lectura y su confiabilidad al momento de la trasmisión de la información.

Contando con tecnologías de digitalización y pensando en el rendimiento óptimo del sistema, las historias médicas serán almacenadas en la base de datos del servidor, logrando así una conexión confiable y fluida que permita la transmisión de archivos de imagen y texto. La selección del método de almacenamiento de las historias médicas en los servidores y los protocolos de transmisión, se harán pensando en esta optimización, lográndose así la eficiencia más alta en la aplicación. Además, se fortalecerá el sistema y se hará menos vulnerable a situaciones hostiles: baja velocidad de transmisión, fallos en la red móvil, alto tráfico, entre otras.

1.II OBJETIVOS

1.II.I OBJETIVO GENERAL

Desarrollar un sistema que permita el acceso a un servidor con historias médicas en una clínica u hospital determinado mediante dispositivos móviles.

1.II.II OBJETIVOS ESPECÍFICOS

- Analizar las distintas tecnologías de digitalización de documentos, para así, poder seleccionar formatos que permitan el fácil manejo en un sistema de transmisión.

- Estudiar el comportamiento de códigos QR, para adaptarlos a las necesidades del sistema y a la aplicación móvil a desarrollar.

- Investigar a fondo estructuras y tecnologías de almacenamiento de información, para poder identificar la más versátil, y la que permita alcanzar una relación costo/eficiencia adecuada.

- Identificar los campos más importantes dentro de una historia médica general, lo que permitirá hacerla objetiva y práctica para el uso de la misma en una sala de emergencias.

- Conocer el comportamiento de las redes celulares y de los servicios WiFi disponibles, para hacer un uso adecuado y efectivo de los mismos.

- Desarrollar una aplicación, para sistema operativo Android, que permita a dispositivos móviles leer e interpretar un código QR, y logre el acceso al servidor.

- Realizar estimado de costo del sistema propuesto.

- Realizar una prueba piloto, para comprobar la factibilidad del sistema y las posibles mejoras que pueda tener.

1.III LIMITACIONES Y ALCANCES

1.III.I ALCANCES

- Este trabajo especial de grado contemplará la creación de un acceso desde dispositivos móviles a un servidor que contenga historias médicas independientemente del formato de ellas. Es decir, cada institución podrá adaptar su formato de historia médica al sistema de acceso al servidor.

- El acceso a las historias médicas se podrá hacer con dispositivos que tengan sistema operativo Android.

- Se logrará una integración del sistema con redes inalámbricas, es decir, el sistema se adaptará al servicio de transmisión de datos disponible (UMTS o WiFi).

1.III.II LIMITACIONES:

- Dadas las dificultades al momento de adquirir el equipamiento necesario para probar distintas tecnologías de identificación, para este estudio se usará solo códigos QR.

- Dada la gran variedad de especialidades y de formatos de historias médicas, este trabajo se limitará solo a información general necesaria en una sala de emergencias, no se especificará información especializada y aplicada a solo un área de la medicina.

- En este trabajo se proveerán soluciones de telecomunicaciones: acceso, confiabilidad de enlace, entre otros. Por lo que cualquier tipo de solución informática, incluyendo el desarrollo de la aplicación Android, se limitará a pruebas y simulaciones que ayuden a comprobar la factibilidad del sistema.

Capitulo 2. Marco Referencial

A continuación, se presentará la información teórica recopilada en la primera fase del proyecto, información teórica que sustenta la aplicación de ciertas estructuras y tecnologías en el desarrollo del sistema.

Tomando en consideración el mapa presentado, se desarrollarán los temas más importantes a ser tomados en cuenta en la elaboración del Trabajo Especial de Grado.

Ilustración 1: Mapa de Conocimientos Necesarios para la Elaboración del TEG

Fuente: Elaboración Propia

2.I Arquitectura Cliente/Servidor

Hoy en día las comunicaciones van más allá de lo que jamás pensamos. La transferencia de datos se ha convertido en la forma preferida de los usuarios al momento de comunicarse. Servicios de correo electrónico y redes sociales, se han convertido en una manera fácil y rápida de compartir información, siendo tan versátiles que ofrecen envío de archivos adjuntos (imágenes, documentos de texto, notas de voz, etc.), lo que hace el intercambio de información entre dos o más usuarios más dinámico y atractivo.

Sin embargo, el manejo de estos datos representa un reto para los programadores y diseñadores, ente la creciente demanda, es por esto que los informáticos crearon estructuras físicas y lógicas que permiten la organización y manejo de toda esta información. Con el paso de los años y el auge del protocolo IP (*Internet Protocol*), grandes y medianas empresas, con la finalidad de ahorrar y optimizar sus sistemas, han decidido usar información en datos para gestionar ciertos documentos. Esto significa que toda esta plataforma debe contar con altos niveles de seguridad. Empresas como **IBM** (*International Business Machines*), han creado una infraestructura de gestión de datos, que permite brindar solución a todos los problemas que puedan presentarse; esta infraestructura, depende básicamente de cuatro renglones; Disponibilidad de Información, Retención de Información, Seguridad de Información y Conformidad de la Información. [1]

Uno de los renglones más importantes de cualquier estructura es la Disponibilidad de Información, ya que esto es lo que hace al sistema funcional, confiable y seguro. Actualmente, en el mundo de la informática existen estructuras que permiten guardar información para hacerla accesible en cualquier momento, existe el almacenamiento físico o *Hard Drives*, que permiten un almacenaje de información limitado por su capacidad, y que son muy usados para gestionar información de pequeño o mediano tamaño.

También existen estructuras lógicas que permiten hacer el mismo trabajo, estas estructuras son accesibles desde ordenadores, PDA, dispositivos móviles, entre otros. Los servidores han venido revolucionando la manera en que guardamos información en la Web. De hecho empresas reconocidas como **APPLE INC**., ofrecen servicios de almacenamiento en nube (*Cloud*), que le permiten al usuario guardar cierta cantidad de información en servidores, con acceso desde cualquier dispositivo con sistema operativo **OSX** [2]. De igual forma, otros sistemas operativos como **UBUNTU 11** le permiten al usuario servicios muy parecidos para almacenamiento en la Web.

Esta nueva forma de compartir información obedece a una arquitectura distribuida Cliente/Servidor, que permite a los usuarios finales, obtener acceso a la información de forma transparente; aun en entornos multiplataforma [3]. En la arquitectura Cliente/Servidor se pueden identificar cuatro niveles importantes: nivel de presentación, nivel de aplicación, nivel de comunicación y nivel de base de datos [4]. En cada uno de estos niveles se gestiona la acción de cada uno de los actores dentro de la arquitectura. El nivel de aplicación gestiona todos los elementos que hacen referencia al cliente. El de presentación, hace referencia a los elementos que involucran al servidor. El nivel de comunicación hace referencia a todo lo asociado con el enlace o *Middleware* y la base de datos.

2.I.I Servidor

Un servidor es todo proceso que proporciona servicio a otros [4]. Se encarga de gestionar todo lo referente al manejo de archivos en la base de datos, traducir y satisfacer las peticiones de los clientes (varios a la vez inclusive), y ser capaz de controlar el tráfico de una plataforma que puede ser multicliente y que además exige varias descargas al mismo tiempo [5].

El tipo de servidor es el que permite dar topología o arquitectura a la red, es decir, dependiendo del tipo de servidor se implementarán protocolos específicos de transmisión y transporte de archivos, se definirán los tipos de formato que se podrán manejar y se establecerán mecanismos de seguridad que generen un flujo de datos confiable y seguro [5].

2.I.II Middleware

El *middleware* es la interfaz que permite al cliente comunicarse con el servidor, ejecutándose en ambas partes [4]. El aporte más importante del *middleware* es la

independencia del servidor del cliente, y viceversa, facilitando así el desarrollo de aplicaciones, sin pensar en las dependencias tecnológicas de las partes. En síntesis, las funciones más importantes del *middleware* son: independizar los entornos propietarios, permitir la conectividad entre los sistemas de información de las partes, crear una conexión más segura al recibir información del cliente y del servidor, y permitir el diseño de arquitecturas basadas en distintas tecnologías en el desarrollo del sistema [5].

2.I.III Cliente

El cliente es un software. También considerado un proceso que permite al usuario final hacer solicitudes al servidor y recibir una respuesta del mismo [4]. Al ser el proceso el que interactúa con el usuario, generalmente posee una interfaz gráfica que le facilita la comunicación con el servidor. Se encarga de generar los comandos de comunicación, y de traducir los comandos recibidos. Además, debe ofrecer edición de archivos descargados y servicios de carga al servidor en forma agradable para el usuario y que cumpla con todas sus necesidades [3].

2.II FTP (File Transfer Protocol)

El protocolo **FTP**, o Protocolo de Transferencia de Datos, es un protocolo alojado en la capa de aplicación del modelo **OSI** (*Open System Interconection*) y del modelo **TCP/IP** [6].

Al momento de haber una comunicación entre dos estaciones (computadores, dispositivos móviles, entre otros) se deben tomar en cuenta varias consideraciones, por ejemplo, el formato del archivo a compartir, los formatos aceptados por cada una de las estaciones, los sistemas de almacenamiento y gestores de directorios de cada

terminal. Estos aspectos generaban inconvenientes al momento de establecer una conversación, por ejemplo en una arquitectura cliente/servidor [6].

El protocolo **FTP** soluciona todos estos inconvenientes agregando una conexión más, a la ya habitual conexión de datos en protocolos de transferencia. **FTP,** además proporciona una conexión de control. Cada una de estas conexiones desempeña un trabajo específico haciendo el sistema mucho más eficiente, en comparación con otros protocolos de la arquitectura cliente/servidor [6].

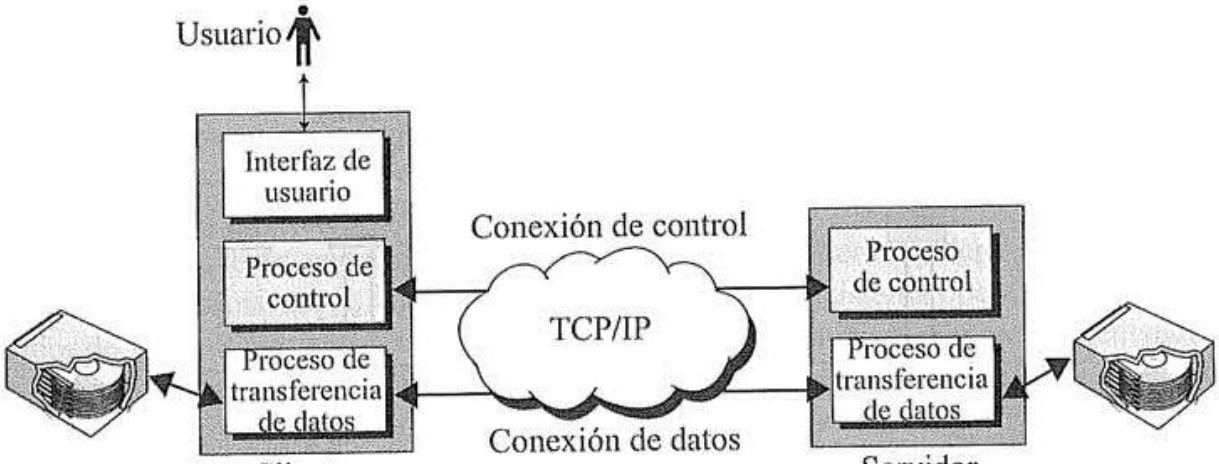

Ilustración 2: Esquema de Protocolo FTP

Fuente: Forouzan

2.II.I Conexiones de Protocolo FTP

2.II.I.I Conexión de Control

La conexión de control se realiza a través del puerto 21. Esta conexión establece la comunicación entre los terminales. Está caracterizada por ser solo de comandos, es decir un terminal establece conexión con otro a través del uso de comandos **FTP**.

Esta conexión es generalmente sencilla, es gestionada a través de los procesos de controles activos en los dos agentes, se mantiene activa todo el tiempo [6].

2.II.I.II Conexión de Datos

Establecida a través del puerto 20. Esta conexión, como su nombre lo sugiere es la encargada de transferir los datos o archivos necesarios. A diferencia de la conexión de control, no se mantiene activa todo el tiempo, sino que se abre cada vez que sea necesario el envío de información. La conexión de datos es gestionada por los procesos de transferencia de datos en los agentes involucrados en la comunicación [6].

Esta conexión presenta una complejidad mayor a la de control, ya que esta debe ser capaz de adaptarse y funcionar con cualquier formato de archivo enviado [8].

2.II.II Arquitectura Cliente/Servidor FTP

El servidor es una estructura informática que se encuentra siempre activa y escuchando cualquier petición que pueda hacer un cliente. El cliente generalmente es un programa o *Software*, que le da al usuario la posibilidad de enviar solicitudes al servidor, y de recibir respuesta del mismo.

Los servidores **FTP**, son programas específicos que permiten la transferencia de datos con uno o varios clientes bajo ciertas condiciones y limitaciones de acceso. Este programa está alojado en una máquina que desde ese momento pasa a llamarse Servidor. Esta máquina debe tener conexión a internet, ya que ese será el medio en el que la estructura Cliente/Servidor haga la transferencia de datos. El servidor debe tener asignada una dirección IP estática, que le permita ser ubicado por los clientes, no importa la circunstancia; en algunos casos se usa un **DNS** (*Domain Name Server*), para darle un nombre a esa dirección estática [6]. Los servidores **FTP** permiten acceso público y privado, esto puede ser bastante útil al momento de dar seguridad a la información que se transmitirá. Para lograr un acceso privado, el servidor permite la creación de cuentas de usuario al que se otorgan beneficios que a usuarios públicos no se les dan. Creando así niveles de confianza en la seguridad del Servidor. A este nivel, el Servidor es capaz de guardar registros de las actividades realizadas por los usuarios, y así conocer el historial de la conexión. (Otras características de la seguridad y funcionamiento del servidor **FTP** se encuentran en la RFC - *Request For Comments* - 959). Para brindar más de seguridad el protocolo **FTP** (capa de aplicación, modelo **TCP/IP**) trabaja sobre puerto 20 y puerto 21 de la arquitectura, para la conexión de datos y la conexión de control. Sobre la capa de transporte trabaja con protocolo **TCP** (*Transmission Control Protocol*), este protocolo es el que conforma el anteriormente nombrado *Middleware*, **TCP**, es lo que permite la comunicación entre el cliente y el servidor [6].

2.II.II.I Servidor FTP

El servidor (d*aemon*), es el encargado de gestionar todas las acciones que sean requeridas por el cliente. El servidor suele ser un *software* alojado en una máquina física, con alta capacidad de almacenamiento, lo que le permite alojar cantidades importantes de información. En principio, el acceso al servidor se hace desde el cliente con una serie de comandos que le exigirán una respuesta o acción [6].

El servidor **FTP** ofrece gran ventaja con respecto a otros servidores ya que, en su diseño y aplicación más pura, está hecho para la transferencia de archivos, sin importar el formato.

2.II.II.II Cliente FTP

El cliente **FTP** es un software multiplataforma que le permite al usuario tener acceso al servidor desde cualquier dispositivo (móvil o fijo) donde se encuentre alojado. El cliente puede ser clasificado dependiendo del tipo de acceso que tenga; el servidor puede ser configurado para dar acceso privado (usuario registrado) o público (anónimo) [6].

2.II.II.II.I Cliente FTP Privado

El cliente de este tipo tiene un nombre de usuario (*username*) y una contraseña (*password*), goza de privilegios de los cuales ningún otro tipo de cliente disfruta, y tiene acceso a toda la información alojada en el servidor. Este usuario es capaz, además de bajar archivos, de subirlos o modificarlos dentro de la base de datos [6].

Este cliente, también tiene la posibilidad de tener un registro de actividades o historial de acceso al servidor, permitiéndole conocer que archivos ha bajado, subido, editado o borrado.

2.II.II.II.II Cliente FTP Público

El cliente público es un cliente anónimo que puede acceder al servidor y bajar información, generalmente a este tipo de usuarios se le limita la cantidad de información a la que puede acceder [6].

Este tipo de cliente no es capaz de subir información al servidor, por lo tanto no podrá bajar un archivo, editarlo y volverlo a subir, esta condición hace que la información dentro del servidor sea más confiable y segura [6].

2.II.III FTP Activo y FTP Pasivo

En una conexión FTP siempre existe un puerto de control y un puerto de transferencia, sin embargo cuáles y quienes determinan estos puertos depende del modo del servidor [7].

2.II.III.I FTP Activo

El modo activo funciona de la siguiente manera:

1. El cliente inicia conexión desde un puerto N aleatorio con el puerto 21 del servidor

2. El cliente escucha las peticiones del servidor en el puerto N+1.

3. Por último es el servidor quien se conecta desde su puerto 20 con el puerto N+1 del cliente.

Esta modalidad no permite al servidor conectarse con el puerto del cliente que escucha si éste se encuentra detrás de NAT (*Network Address Translation*). Para solucionar esto está el modo pasivo [7].

2.II.III.II FTP Pasivo

El modo pasivo funciona de la siguiente manera:

1. El cliente inicia la conexión desde un puerto N aleatorio con el puerto 21 del servidor.

2. El servidor responde informándole al cliente en que puerto estará escuchando sus peticiones.

3. El cliente inicia la conexión de datos desde el puerto N+1 al puerto especificado anteriormente por el servidor [7].

2.II.IV Seguridad (SSL/TLS)

Una manera en que el FTP puede proteger la información que se comparte y verificar que clientes se conectan al servidor es a través de protocolo SSL/TLS (*Secure Sockets Layer/Tranport Layer Security*) [8].

SSL/TLS es un protocolo de seguridad utilizado para resguardar la privacidad de los mensajes en internet, mediante el uso de certificados digitales.

Los tres puntos en los que este protocolo se enfoca son:

- Autentificación del cliente: asegura que el cliente pueda identificar de manera segura al servidor.

- Encriptado de data: la data transferida es encriptada con complejos algoritmos, haciéndola robusta ante cualquier ataque o intercepción.

- Chequeo de Integridad de la Data: Verifica que no haya habido alteración de la data durante la transmisión.

El cliente envía una solicitud que desea hacer una comunicación segura con el servidor, y le envía los datos del protocolo SSL/TLS que soporta. El servidor recibe el pedido y responde informando que está de acuerdo con hacer la conexión. Una vez que ambas partes están de acuerdo en que quieren hacer una comunicación segura, el servidor le envía al cliente su certificado digital [9].

El cliente recibe el certificado y verifica que esté íntegro (utilizando la firma digital), que esté vigente y comprueba que la AC (Autoridad Certificador) sea confiable.

Una vez concluidos estos pasos se tiene una conexión segura.

2.III Códigos QR (*Quick Response Barcodes*)

Los códigos **QR** (*Quick Response Barcodes*) son una tecnología, desarrollada por la empresa japonesa Denso Wave [10], la cual permite crear etiquetas que almacenan información de un objeto con el fin de identificarlo.

Estos códigos consisten en un arreglo bidimensional de módulos de cuadros, los cuales están ordenados en un cuadro más grande [10]. La data se representa en código binario, donde el 1 se ve como un cuadro negro y el 0 como un cuadro blanco.

Ilustración 3: Código QR
Fuente: Denso Wave

Son capaces de manejar distintos tipos de data: Numérico, alfanumérica y caracteres. Un solo código es capaz de almacenar un máximo de 7089 caracteres numéricos y 4296 alfanuméricos.

No solo permiten almacenar grandes cantidades de información sino que lo hacen en tamaños reducidos.

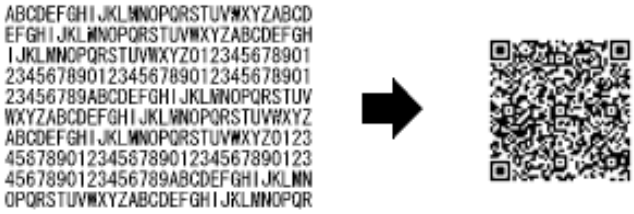

Ilustración 4: Almacenamiento en Código QR

Fuente: Denso Wave

La capacidad de almacenamiento depende del tamaño de la etiqueta; es decir, del número de módulos. Los tamaños se identifican con versiones. Éstas van desde la versión 1 (21x21 módulos) hasta la 40 (177x177 módulos), cada vez que se aumenta de versión, el número de módulos aumenta en 4.

Los códigos **QR** permiten la corrección de errores, lo cual los hace resistentes a daños. Existen varios niveles de corrección:

- L: corrige un 7% del total de bits.
- M: corrige un 15% del total de bits.
- Q: corrige un 25% del total de bits.
- H: corrige un 30% del total de bits.

El nivel de corrección de error también influye en la capacidad de almacenamiento. Éste es elegido dependiendo del ambiente en el que se encuentre el objeto identificado. Si existe una probabilidad de roce o desgaste relativamente alta, se elige el nivel con mayor corrección. Sin embargo, seleccionar un nivel alto, hace que la cantidad de data que se pueda almacenar se vea reducida [11].

Versión	Nivel de Corrección	Numérico	Alfanumérico
1	L	41	25
	M	34	20
	Q	27	16
	H	17	10
2	L	77	47
	M	63	38
	Q	48	29
	H	34	20
3	L	127	77
	M	101	61
	Q	77	47
	H	58	35
4	L	187	114
	M	149	60
	Q	111	67
	H	82	50
.	.	.	.
.	.	.	.
.	.	.	.
38	L	6479	3927
	M	5039	3954
	Q	3599	2181
	H	2735	1658
39	L	6743	4087
	M	5313	3220
	Q	3791	2298
	H	2927	1774
40	L	7089	4296
	M	5596	3391
	Q	3993	2420
	H	3057	1852

Tabla 1: Nivel de Corrección de Error

Fuente: Denso Wave

2.IV Android

Android es un sistema operativo de nueva generación y de código abierto, bajo lenguaje de JAVA, desarrollado por **Google** y **OHA** (*Open Handset Alliance*), basado en **LINUX** en su núcleo, tomando de este la capacidad de manejar memoria y procesos, gestión de seguridad y controladores de *hardware* [12].

Hasta ahora es un sistema operativo exclusivo de dispositivos móviles (*smartphones y tabletas*), buscando así la integración de servicios que permitan convertir a este tipo de dispositivo en una herramienta del día a día, que facilite las tareas del usuario [12].

La principal ventaja de este sistema operativo en comparación con otros, es la libertad que ofrece al momento de desarrollar aplicaciones, ya que proporciona un entorno abierto de programación, y además ofrece un mercado de aplicaciones que permite comercializar aplicaciones de cualquier usuario o programador [13].

Como ya se mencionó, en su núcleo cuenta con una estructura basada en LINUX. Sin embargo, Android se complementa con una arquitectura en capas, en la que cada una desarrolla una o varias tareas en específico.

Tal y como se presentan en la Ilustración 5.

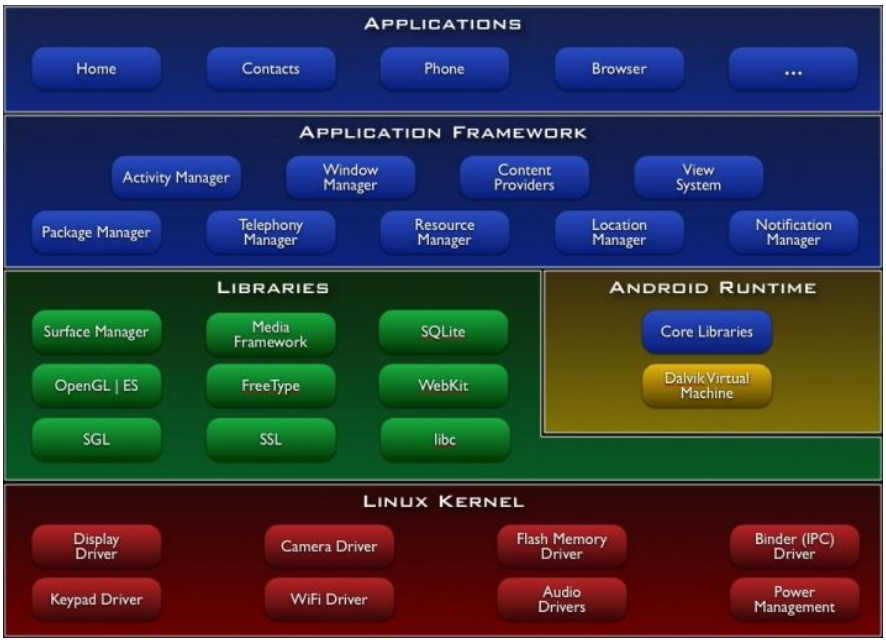

Ilustración 5: Arquitectura Android

Fuente: Android Developers

El núcleo (*kernel*) es el encargado de manejar drivers (audio, WiFi, pantalla), gestionar el manejo de memoria, entre otros.

Las librerías del sistema operativo residentes en la capa librerías, son las encargadas de dar a la aplicación la capacidad de manejar documentos, bases de datos, reproducción de formatos específicos de videos, y entre otras cosas brindar seguridad y herramientas necesarias para el óptimo funcionamiento de cualquier contenido visible en el dispositivo [13].

Por otro lado está el espacio de trabajo de las aplicaciones, aquí es donde se le presentan al programador las posibilidades y los permisos necesarios para acceder a cualquier parte del *hardware* que considere necesaria para el funcionamiento de la aplicación [13].

Por último se tienen las aplicaciones, éstas son el producto final, totalmente basadas en lenguaje de programación JAVA, es la interfaz encargada de gestionar todas las peticiones del usuario dentro de una aplicación y brinda oportunidades casi infinitas en su diseño [14].

El entorno de programación de Android, Eclipse, brinda todas las posibilidades del entorno de programación JAVA, poniendo a disposición del programador los JDK (*Java Development Kits*), además los SDK (*Software Development Kits*) orientados a Android.

Los JDK, son programas que brindan herramientas para la programación bajo lenguaje de JAVA, entorno de desarrollo y compiladores, entre otros.

Los SDK, son programas que le dan al programador todas las herramientas necesarias para correr su código JAVA en emuladores de sistema operativo Android, estos emuladores son llamados AVD (*Android Virtual Device*), y requieren una configuración específica, ya que pueden correr bajo cualquier versión del sistema operativo, y bajo cualquier dispositivo emulado (teléfono o tableta) [14].

Capitulo 3. Marco Metodológico

Para la realización de este Trabajo Especial de Grado, se propuso una metodología organizada en fases, que ayudaron a dividir el proyecto para facilitar su desarrollo. Cada una de estas 4 fases representa un 25% del proyecto total y definitivo, entre ellas existe una relación de dependencia y prelación. La siguiente dependerá enteramente del cumplimiento de la anterior. Desde la investigación teórica para formar bases hasta la implementación total del sistema, las siguientes 4 fases describen de manera específica la estructura, paso a paso del desarrollo del Trabajo Especial de Grado.

3.I Fase 1: Investigación Teórica y Recopilación de Información

El desarrollo de esta fase se basó enteramente en investigación teórica. Se llevó a cabo un estudio exhaustivo de tecnologías de escaneo y digitalización de documentos e imágenes, estructuras de almacenamiento de información y del lenguaje de programación JAVA orientado a Android (este último con el soporte de la electiva cursada). También se estudiaron las historias médicas desde el punto de vista de su estructura y contenido, extrayendo de ellas lo más importante en una sala de emergencia.

3.II Fase 2: Análisis de Información y Toma de Decisiones

En esta fase del proyecto, con el análisis obtenido de la investigación teórica realizada, se tomaron decisiones y se puntualizó en tecnologías y métodos a utilizar en el desarrollo del sistema. Una vez identificados todos los factores y aspectos a considerar, se procedió con las primeras pruebas y simulaciones que arrojaron resultados que ayudaron a mejorar la implementación del sistema en un futuro.

3.III Fase 3: Diseño y Simulación

Con los resultados obtenidos de las simulaciones se tomaron decisiones que ayudaron a diseñar la topología final del sistema, asegurando así, su versatilidad y adaptabilidad a las tecnologías de trasmisión de datos móviles. También se realizó una estimación de costos, que demuestra que, en el ámbito económico este proyecto es factible y conveniente.

3.IV Fase 4: Implementación

La etapa final del proyecto fue la implementación, en la cual se comprobó de forma práctica, que el sistema es factible desde el punto de vista técnico y económico, cumple con las funciones requeridas y con las características planteadas: seguridad, disponibilidad, movilidad, versatilidad y manejabilidad; logrando así el objetivo general de este Trabajo Especial de Grado. Adicionalmente se hizo una prueba piloto para probar el sistema en un ambiente real.

Capitulo 4. Desarrollo

En el siguiente capítulo se detallarán los pasos seguidos para la configuración, establecimiento y control de todas las tecnologías y estructuras usadas en el desarrollo de este Trabajo Especial de Grado. Para facilitar su estudio, se presentarán figuras (capturas de pantallas y fotos) que ayuden al completo y claro entendimiento del funcionamiento del sistema.

4.I Códigos QR

Los generadores de códigos QR pueden ser encontrados y descargados gratuitamente de la red. Para este proyecto se utilizó *Quantum QR Code Generator* el cual puede ser descargado del siguiente link [15].

Una vez que se descargó el programa se procedió a su instalación. El primer paso fue abrir el ejecutable y se abre la ventana de bienvenida.

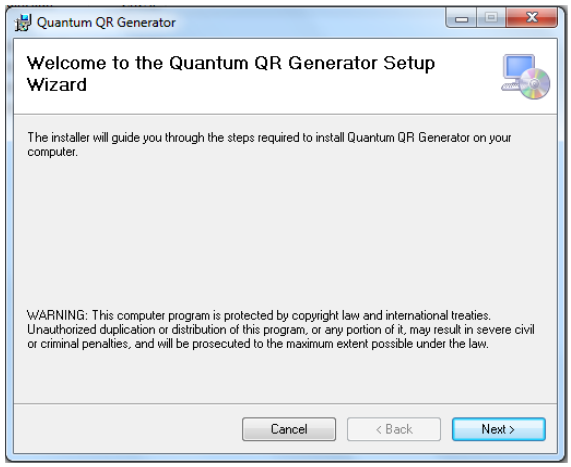

Ilustración 6: Instalador Quantum QR

Fuente: Elaboración Propia

En la siguiente ventana se seleccionó la carpeta donde se deseaba instalar el programa y se le da la opción al administrador para que el programa sea de uso privado o público. Para brindar más seguridad al proyecto se procedió a usar la opción del privado. Una vez concluido este paso el generador queda instalado.

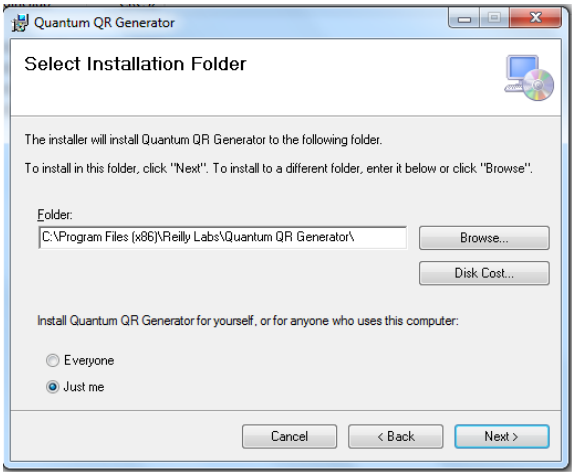

Ilustración 7: Selección de Carpeta de Destino de Instalación de Quantum QR Generator

Fuente: Elaboración Propia

El generador funciona de la siguiente forma:

Cuando el programa se abre aparece la siguiente interfaz:

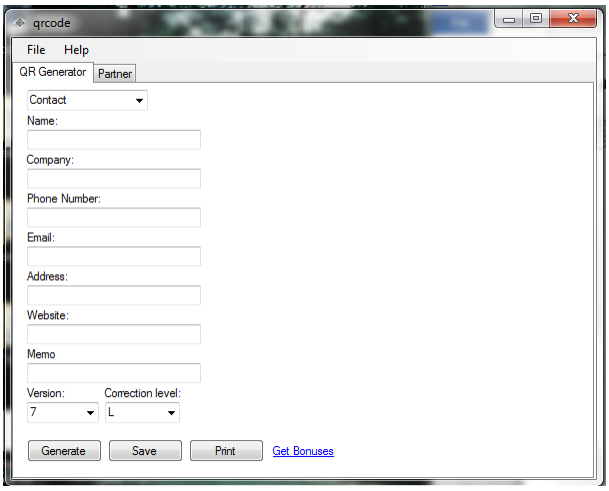

Ilustración 8: Interfaz Principal Quantum QR Generator

Fuente: Elaboración Propia

En esta pantalla aparecen distintos campos. El primero indica el tipo de información que va a ser impresa en el código QR (Contacto, Texto, SMS, URL, Correo electrónico y Teléfono). En el inferior de la pantalla se encuentran las opciones de versión y el nivel de seguridad del código generado.

Para esta aplicación en el recuadro de tipo de información se eligió la de ***Text.***

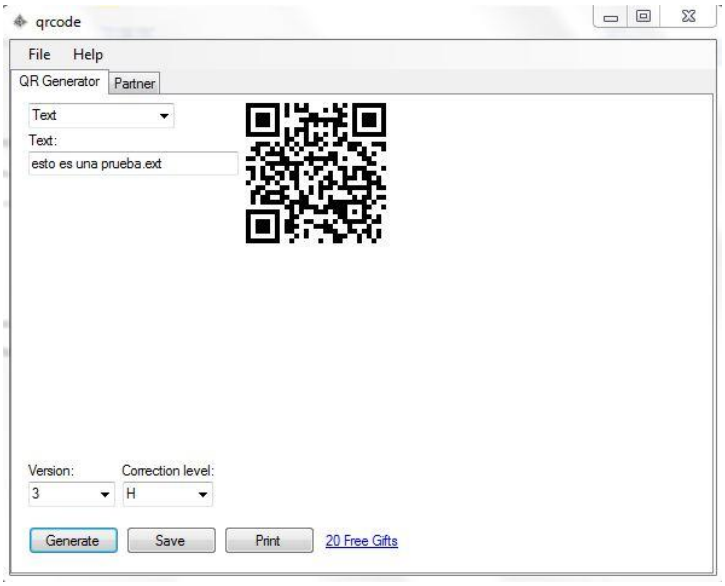

Ilustración 9: Código QR Generado

Fuente: Elaboración Propia

En el recuadro ***Text*** se coloca lo que se quiere imprimir.

4.II Servidor FTP

Una de las partes más importantes de este proyecto, es la elección de servidor FTP. Éste debe ser práctico, seguro, sencillo de utilizar e instalar. Para poder cumplir con estas características se seleccionó el servidor *FileZilla* de Microsoft. Éste es un *software* libre y de código abierto que puede ser fácilmente encontrado y descargado de la Web a través del vínculo [16].

La primera etapa para su instalación fue la descarga de la versión del servidor. A continuación se mencionan los pasos que se siguieron para su instalación:

1. Se seleccionó el tipo de instalación que se deseaba, en el caso de este proyecto, se eligió el "***Custom***" (personalizada). De esta manera se pudo instalar y configurar el servidor con los requerimientos necesarios para este proyecto.

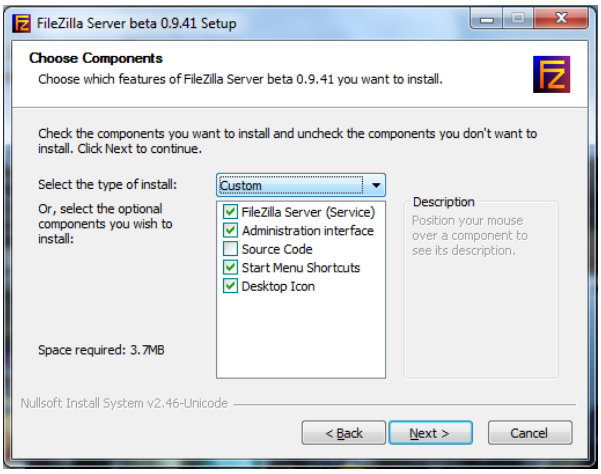

Ilustración 10: Tipo de Instalación del Servidor *FileZilla*

Fuente: Elaboración Propia

2. Se seleccionó la dirección de la carpeta donde se deseaba instalar el servidor

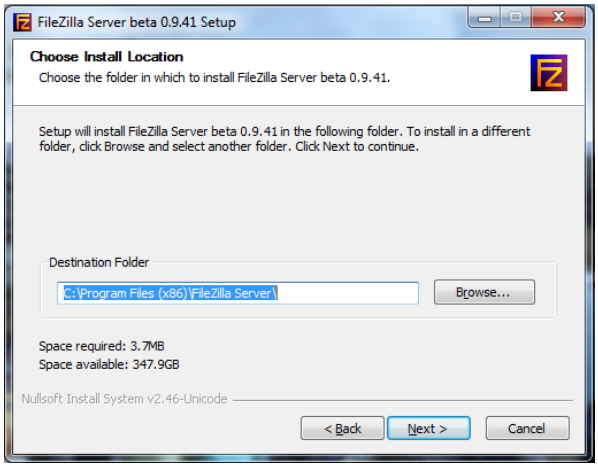

Ilustración 11: Selección de la Carpeta Destino de Instalación del Servidor *FileZilla*

Fuente: Elaboración Propia

3. En este tercer paso se presentaron las distintas alternativas de cómo se puede arrancar el servidor:

- *Install as Service, started with Windows (default):* el servidor se abre al momento de iniciar Windows.

- *Install as service, started manually*: el servidor se abre de forma manual.

- *Do not start as service, start server automatically (not recommended)*

Se eligió la **primera** opción debido a que ésta abre el servidor automáticamente cuando se inicia Windows, evitando así que el administrador lo tenga que abrir cada vez que enciende la computadora. Al abrirse el servidor el administrador lo único que debe hacer es ingresar la clave.

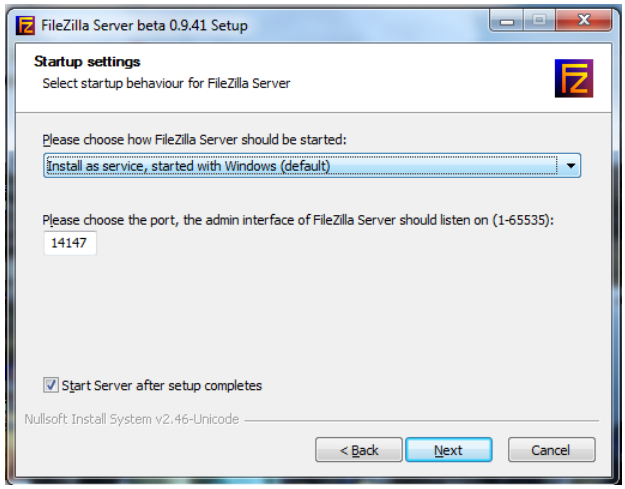

Ilustración 12: Configuración de Instalación del Servidor *FileZilla*

Fuente: Elaboración Propia

4. En este paso se eligió como se deseaba instalar la interfaz del servidor.

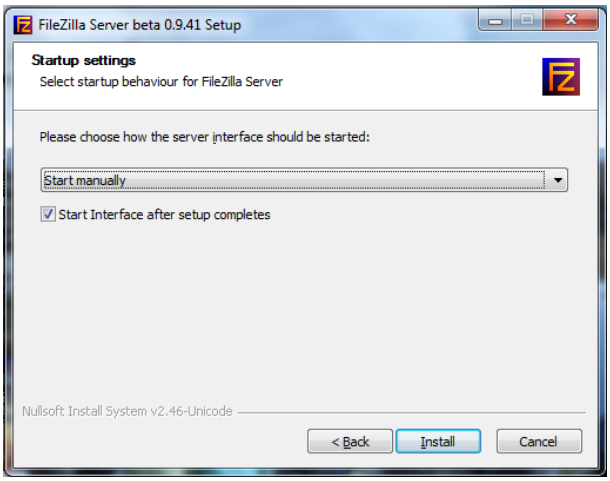

Ilustración 13: Configuración de Instalación del Servidor *FileZilla*

Fuente: Elaboración Propia

Para la segunda etapa se muestra la interfaz gráfica del servidor. En esta pantalla aparecen descritas las acciones y las configuraciones del servidor.

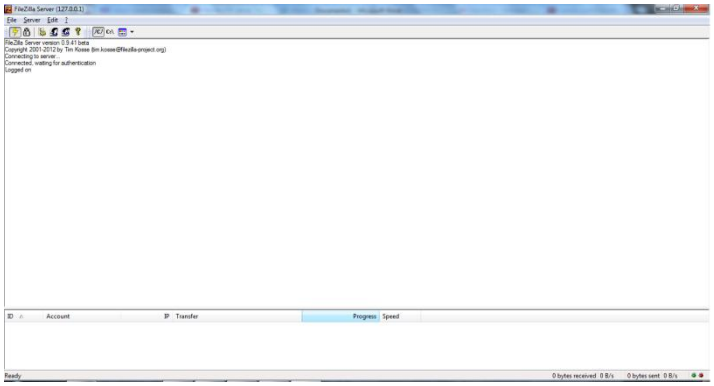

Ilustración 14: Interfaz Gráfica del Servidor *FileZilla*

Fuente: Elaboración Propia

Una vez instalado en la computadora el servidor, se procedió a configurar sus opciones de seguridad y acceso.

1. Lo primero que se configuró fue la clave de acceso del administrador para poder ingresar.

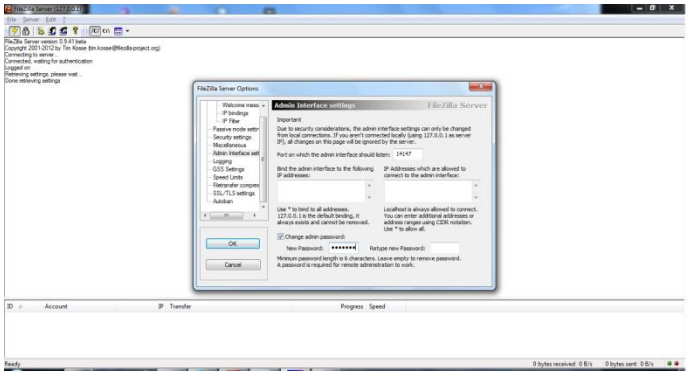

Ilustración 15: Configuración del Administrador del Servidor *FileZilla*

Fuente: Elaboración Propia

2. Se creó el grupo que va poder utilizar el servidor. Los grupos son nombres bajo los que se pueden agrupar un número de usuarios, y configurar los accesos debidos. De esta manera se puede trabajar con diferentes grupos de personas con distintos requerimientos y niveles de permisos.

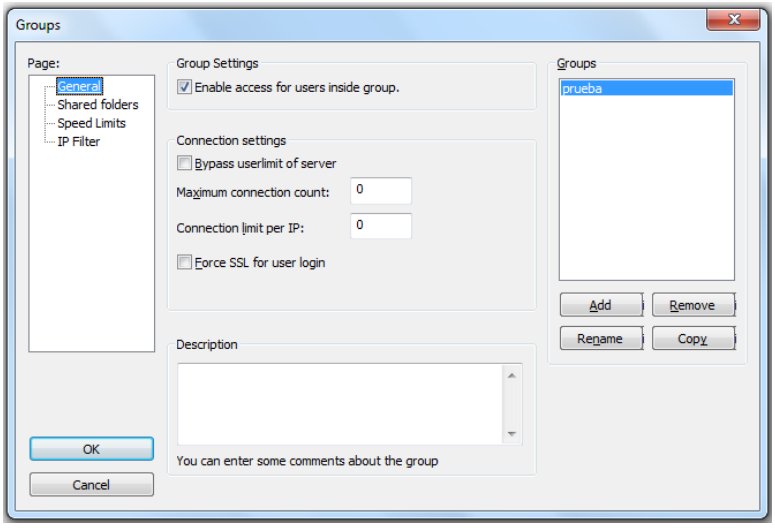

Ilustración 16: Configuración de Usuarios del Servidor *FileZilla*

Fuente: Elaboración Propia

3. Se eligió la carpeta que se va a compartir en el grupo. También se configuraron las posibles acciones que el usuario va a poder realizar sobre los archivos (Leer, editar y borrar).

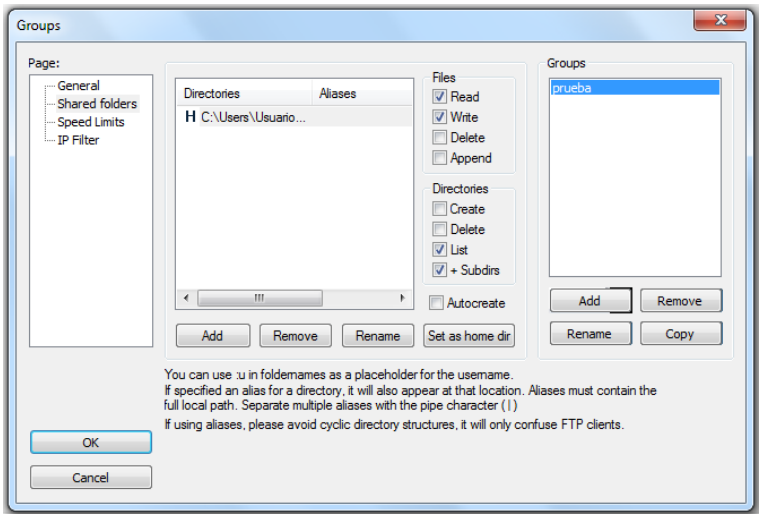

Ilustración 17: Configuración de Carpeta Compartida del Servidor *FileZilla*

Fuente: Elaboración Propia

2.4 Se crearon los distintos usuarios que tienen acceso a las carpetas. A cada usuario se le otorga una clave para poder conectarse al servidor.

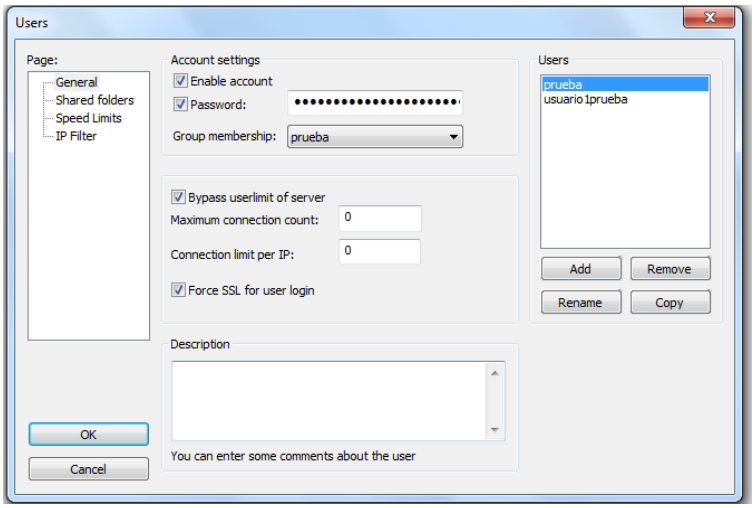

Ilustración 18: Configuración de Grupos del Servidor FileZilla

Fuente: Elaboración Propia

5. Una manera de evitar que virus o gusanos entren al servidor fue habilitando la opción de *Autoban*. Esta opción configura al servidor para que después de un número fallido de intentos para ingresar a éste, se genere una prohibición de entrada automática.

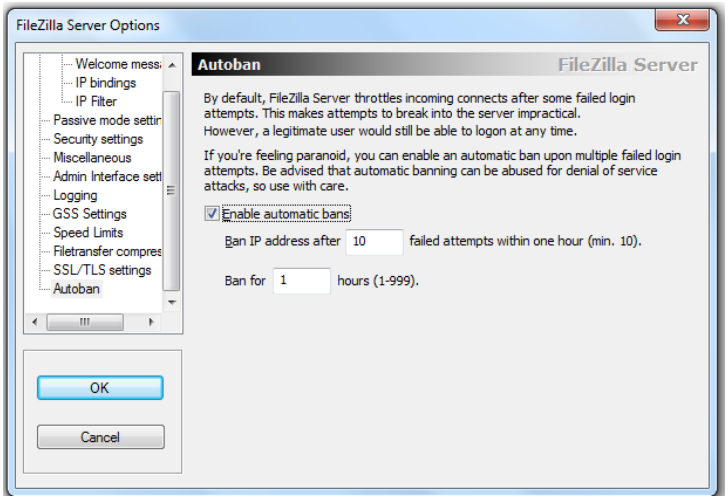

Ilustración 19: Configuración *Autoban* del Servidor *FileZilla*

Fuente: Elaboración Propia

6. Este servidor también cuenta con certificados de configuración SSL (*Secure Sockets Layer*) y TLS (*Transport Layer Security*). El primer paso para generar estos certificados fue colocar los datos necesarios para generar la llave privada.

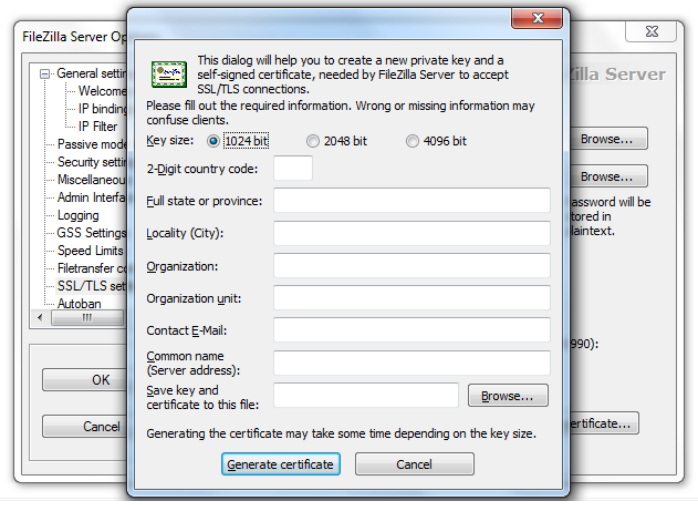

Ilustración 20: Generador de Certificados del Servidor *FileZilla*

Fuente: Elaboración Propia

7. Se ubicó el directorio donde se guardó el certificado y se asignó una clave.

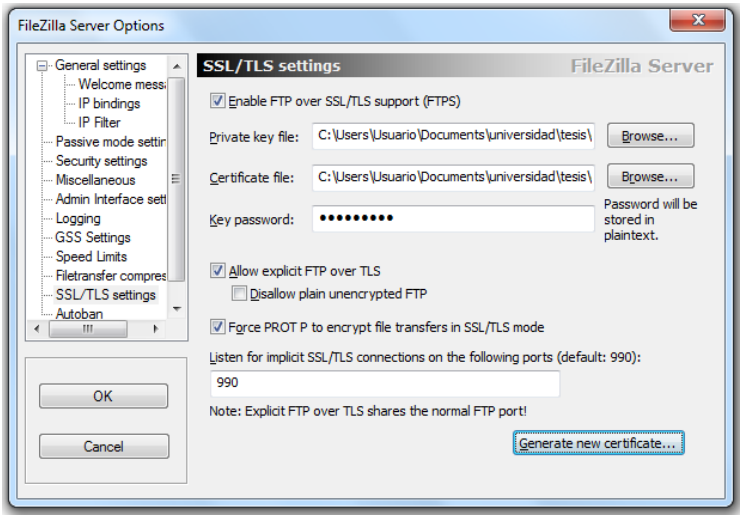

Ilustración 21: Certificado del Servidor *FileZilla*

Fuente: Elaboración Propia

Para poder utilizar el servidor FTP era necesario habilitar los puertos 20 y 21, así como también el rango de aquellos con los que se deseaba trabajar.

Para lograr esto se siguieron los siguientes pasos:

Primero se deben abrir los puertos que se desean utilizar. Para esto nos dirigimos a Panel de control, Sistemas de Seguridad, Firewall de Windows, Configuración Avanzada, reglas de entrada.

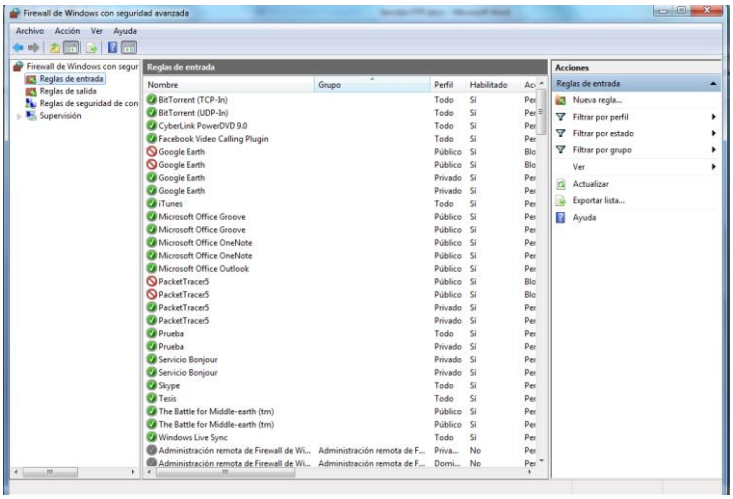

Ilustración 22: Reglas de Entrada en Windows 7

Fuente: Elaboración Propia

Se selecciona la opción de Nueva regla. Se presentan tres opciones, se elige la de puerto.

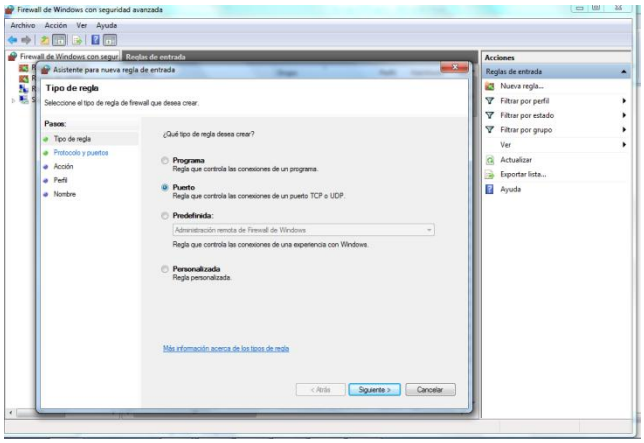

Ilustración 23: Configuración de Puertos en Windows 7

Fuente: Elaboración Propia

Se selecciona TCP o UDP y se elige los puertos que se desean abrir.

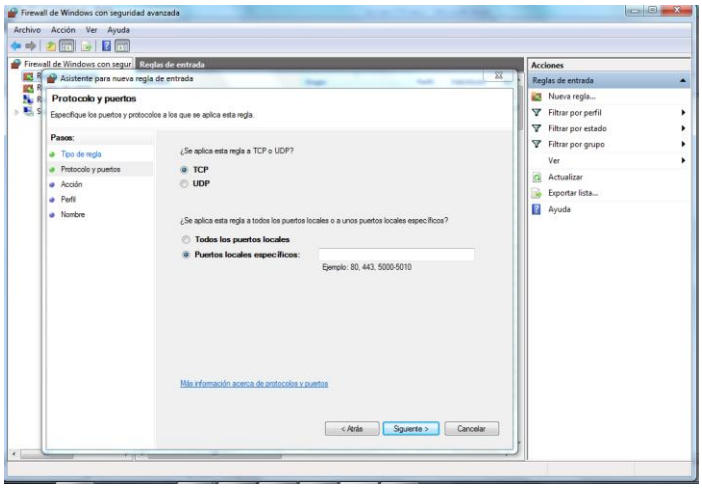

Ilustración 24: Configuración de Puertos de Entrada en Windows 7

Fuente: Elaboración Propia

Una vez que estén abiertos los puertos, se procede a hacer el proceso de "*Forwarding*" en el router de la red.

El primer paso es ingresar a la configuración del router de la red, para esto se coloca la dirección de la puerta de enlace predeterminada en un navegador de internet.

Ilustración 25: Interfaz Gráfica Router Linksys

Fuente: Elaboración Propia

Se escribe los puertos a los que se le desea hacer *forwarding* y la dirección IP del servidor.

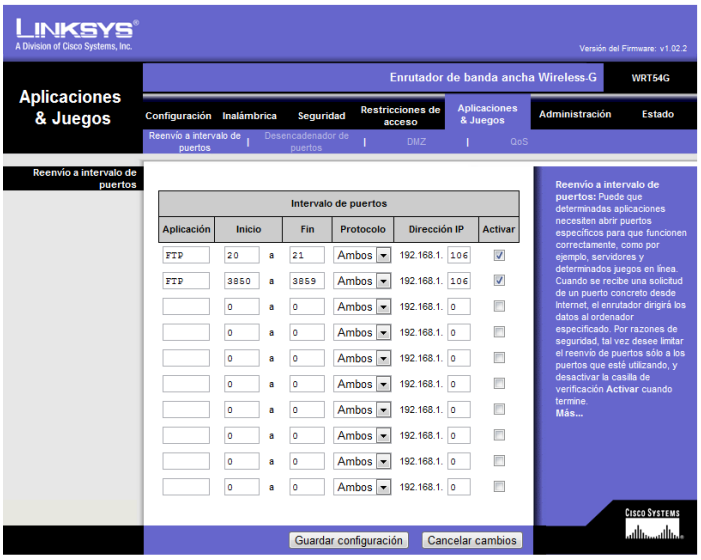

Ilustración 26: *Port Forwarding* en el Router Linksys

Fuente: Elaboración Propia

4.III Aplicación Android

Antes de empezar a desarrollar la aplicación hubo que preparar el ambiente adecuado para aplicaciones JAVA. Este ambiente se empezó a configurar instalando los JDK. El paquete de JDK incluye también el JRE lo que permitió a las aplicaciones desarrolladas compilar y funcionar en la computadora. Estas herramientas fueron descargadas directamente de la página de internet de ORACLE.

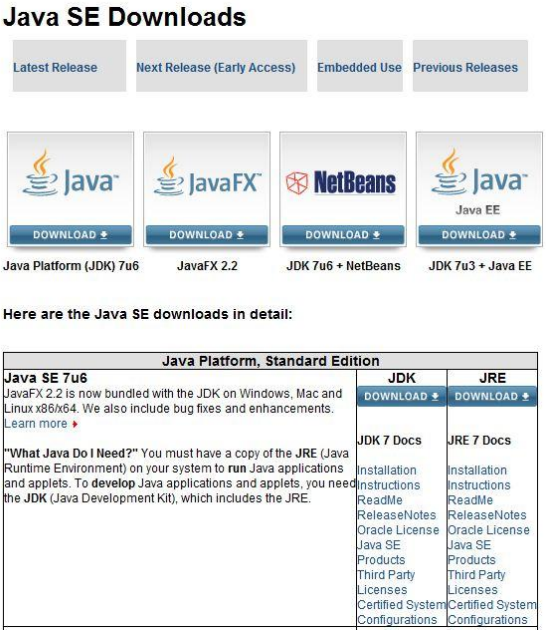

Ilustración 27: Versión de Descarga de los JDK

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

Una vez descargada e instalada las herramientas JDK y JRE, se procedió a acondicionar el ambiente de desarrollo para trabajar con Android, para esto fue necesario descargar los SDK, los cuales permitieron adaptar las herramientas JAVA para el desarrollo de aplicaciones Android. Los SDK se descargaron de la página oficial de Android.

Ilustración 28: Versión de Descarga del Android SDK

Fuente: http://developer.android.com/sdk/index.html

Al adaptar las herramientas de JAVA para desarrollar aplicaciones Android, fue indispensable descargar el compilador, el cual no solo permitió desarrollar los códigos de la aplicación, sino también procesarlos. Este compilador se llama ECLIPSE, y se descargó de la página oficial del producto. La descarga correspondió al tipo de sistema operativo donde se alojó el compilador.

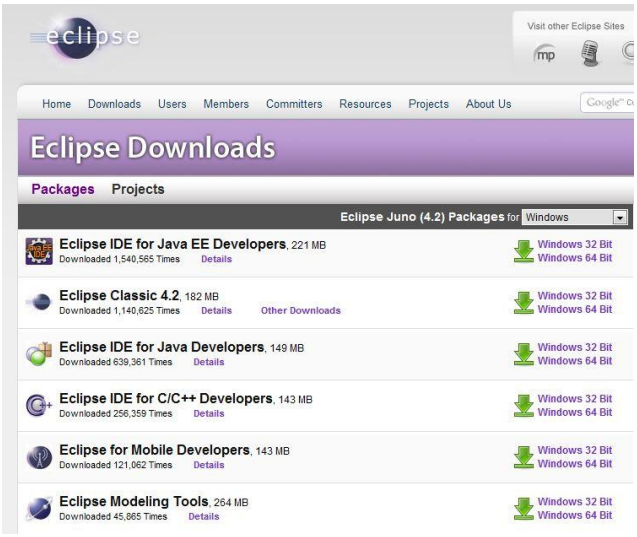

Ilustración 29: Descarga del Entorno de Desarrollo Eclipse

Fuente: http://www.eclipse.org/downloads/

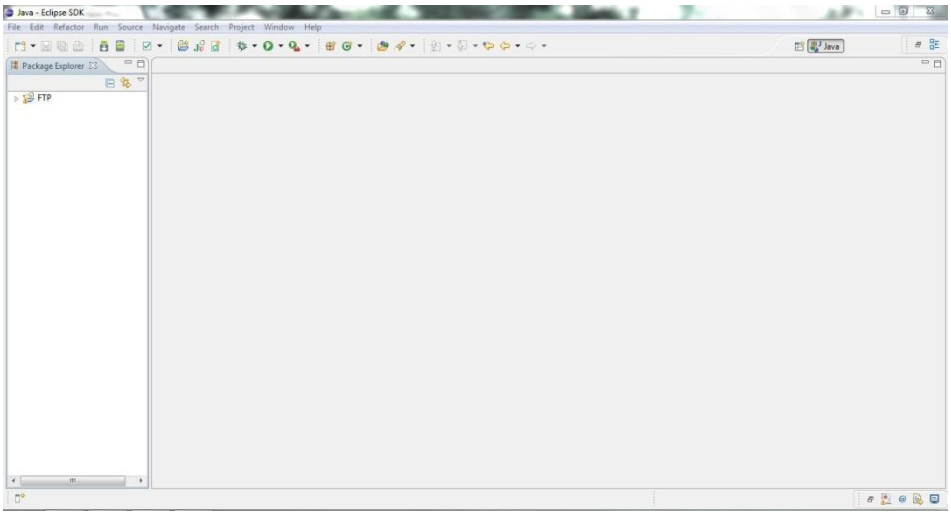

Ilustración 30: Interfaz Gráfica del Entorno de Desarrollo Eclipse

Fuente: Elaboración Propia

Una vez instalado el compilador fue necesario instalar el ADT, lo que permitió al compilador adaptarse al código y librerías nativas de Android. Para instalar el ADT fue necesario ir a la ventana "*Install New Software*", en la cual se le indicó al programa la dirección necesaria para descargar el ADT e instalarlo.

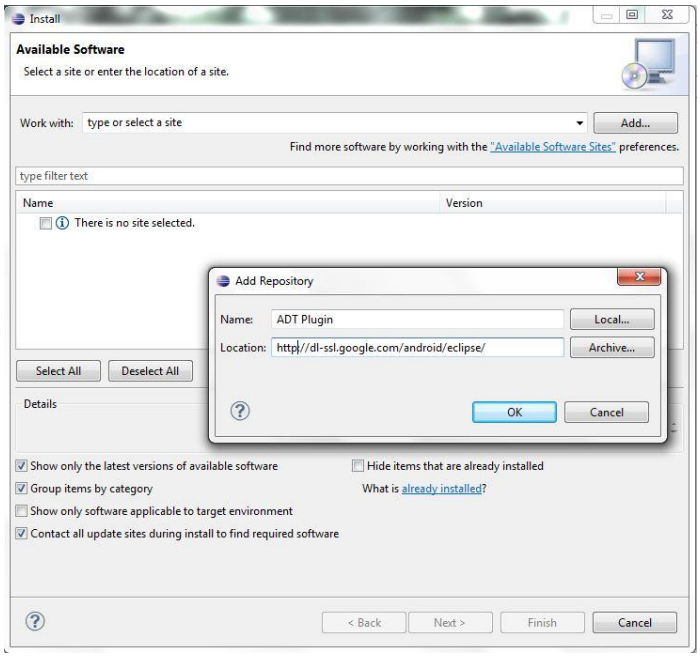

Ilustración 31: Instalación del ADT Plugin
Fuente: Elaboración Propia

Hecho esto, tuvimos disponible en el programa el "SDK Manager", lo que permitió manejar, actualizar, instalar o desinstalar versiones y paquetes necesarios para cada aplicación que se deseaba desarrollar.

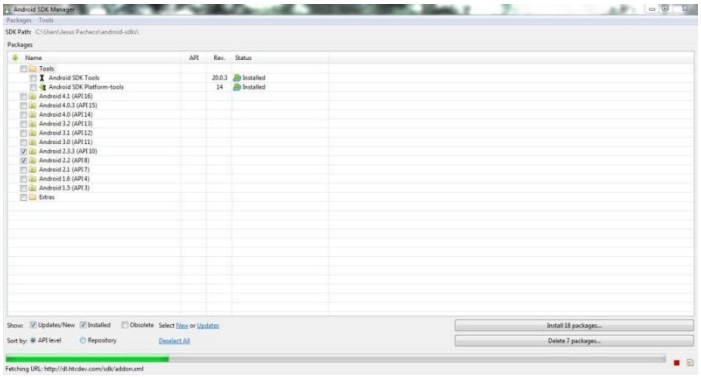

Ilustración 32: SDK Manager de Eclipse
Fuente: Elaboración Propia

Habiendo tenido listo el ambiente de desarrollo, se empezó a escribir el código de la aplicación.

Antes de empezar a escribir el código, se hizo una planificación, en la cual se acordó entre otras cosas la estructura de la aplicación y el diseño de los *layouts.*

Para el desarrollo de la aplicación se decidió usar la librería *Commons Net* 2.0, que dentro de sus paquetes cuenta con el org.apache.commons.net.ftp. Este paquete contiene a la clase FTPClient, que a su vez contiene métodos que permiten la interacción total con clientes y servidores FTP y SFTP.

También se usó una **Instancia** o *Intent* a la librería ZXING, librería de Google que usa la aplicación *Barcode Scanner* de ZXING para leer e interpretar códigos QR de cualquier tipo. Es por esto que es estrictamente necesario que en el equipo estuviese disponible esta aplicación.

Antes de empezar con la escritura del código de la aplicación, fue necesario importar estas librerías al ECLIPSE, para que este fuese capaz de usar sus métodos sin generar errores. Primero se importaron los archivos con extensión .jar, y luego se indicó a la aplicación que los utilizase como librerías de referencia.

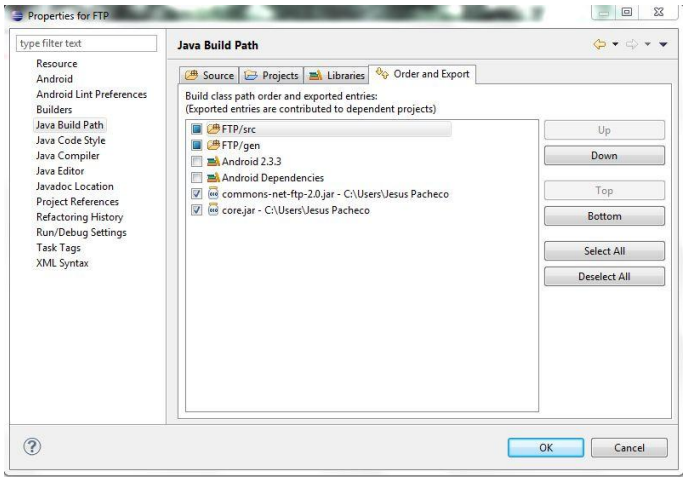

Ilustración 33: Importaciones de Librerías a Eclipse
Fuente: Elaboración Propia

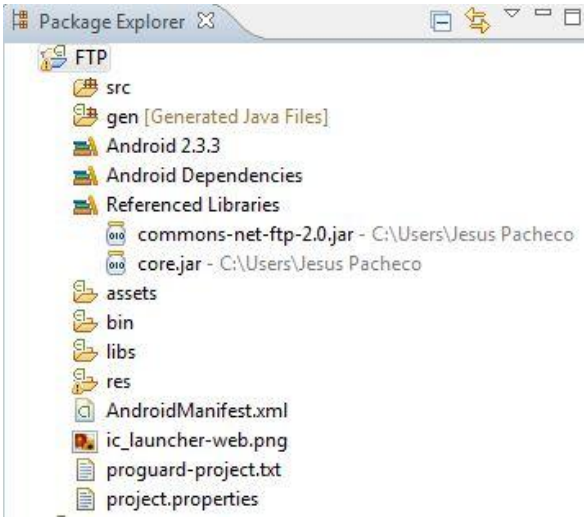

Ilustración 34: Librerías Importadas a Eclipse para su uso en la Aplicación
Fuente: Elaboración Propia

Una que vez se hicieron las importaciones necesarias se pudo empezar a desarrollar el código. Según la planificación hecha previamente, el primer *layout* de la aplicación consiste en un mensaje de bienvenida y un menú que presenta dos botones, que ofrecen al usuario las opciones de cargar o descargar un archivo, según sea su necesidad.

A nivel de código fuente, lo primero que se debió hacer fue declarar nuevos objetos de tipo Botón, que fueron configurados para ser presionados y llevar al usuario a un nuevo *layout* que tiene características según sea la opción. Para programar esto nos valimos de *Intents* que siguiendo la orden de una estructura condicional hacen el direccionamiento requerido.

```
public class FTP extends Activity implements OnClickListener
{
    Button cargar, descargar;
    Intent carga, descarga;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.opciones);

        cargar = (Button)findViewById(R.id.button1); //se crean los objetos de tipo boton
        descargar = (Button)findViewById(R.id.button2); //cargar y descargar

        cargar.setOnClickListener(this);    //los objetos son configurados para
        descargar.setOnClickListener(this); //activar el metodo al ser presionados
    }//oncreate

    public void onClick(View v)                             //al ser presionados, los objetos generan un metodo resultado,
    {                                                       //en este caso se condiciona para discriminar que boton fue
        if (v == cargar)                                    //presionado, cada uno instanciara a una clase diferente
        {
            carga = new Intent(FTP.this, Cargar.class);         //instancia a la clase Carga
            startActivity(carga);
        }//if cargar

        if (v == descargar)
        {
            descarga = new Intent(FTP.this, Descargar.class);   //instancia a la clase Descagarga
            startActivity(descarga);
        }//if descargar
    }//onclick
}//ftp
```

Ilustración 35: Código Fuente Menú de Opciones
Fuente: Elaboración Propia

Dependiendo de la opción seleccionada se muestra un *layout*. En este caso suponiendo que se presiona la opción de Cargar, inmediatamente se puede ver un *layout* con una lista de todos los documentos guardados en la carpeta de historias médicas existente en la memoria externa del teléfono o *SDCard* (para ilustración se creó una carpeta llamada Historias).

A nivel de código, se creó un *layout* de tipo *ListView*. Este tiene la capacidad de mostrar elementos uno debajo de otro, y hacer una vista deslizable hacia arriba o abajo según sea necesario. Para poder mostrar un *layout* de este tipo, fue necesario antes llenar un adaptador llamado *ListAdapter*. Solo adaptadores de esta clase son aceptados por este tipo de *layout*. Este adaptador es una analogía a una pila o cola que se llena con elementos de un arreglo de archivos.

Primero se creó una lista, que se llena con un arreglo de los elementos contenidos en la *SDCard*. Esta lista sirve para crear un vector de tipo archivo (File[]), que posteriormente es cargado en el adaptador de tipo *ListAdapter*. Una vez lleno este adaptador se configuró para ser mostrado en el *layout* correspondiente, de esta forma; en cada fila de la lista se muestra el nombre de cada uno de los archivos que contiene la carpeta Historias en la memoria externa del teléfono.

```
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);

    historias = new ArrayList<String>();    //se crea una lista donde seran guardadas las historias
    directorio = new File(Environment.getExternalStorageDirectory().toString()+"/Historias"); //ruta a la sdcard
    arreglo = directorio.listFiles(); //se crea un arreglo de archivos que se llenara con los elementos en la sdcard

    if(arreglo.length>0)                    //llenado del arreglo
        {
            for( File f:arreglo)
            {
                historias.add(f.getName());
            }//for
        }//if

        adaptador= new ArrayAdapter<String>(this, R.layout.cargar, historias); //se crea y se muestra el adaptador
        setListAdapter(adaptador);                                              //de listview, para mostrar la lista en pantalla
}//oncreate
```

Ilustración 36: Código Fuente *ListView*
Fuente: Elaboración Propia

Una vez mostrada la lista, la forma de cargar un archivo al servidor es sencillamente presionando el nombre del archivo que se desea transmitir. Inmediatamente, esto desencadena un método que usando las instrucciones necesarias carga el archivo a la carpeta adecuada en el servidor e indica a través de un mensaje en pantalla si el archivo fue transmitido correctamente o no.

En el código fuente esto se logró implementando un método que convierte a todos los elementos de la lista en "botones" esperando ser presionados. Cuando esto pasa el "método" que desencadena guarda la posición que fue presionada, el nombre de la etiqueta y la dirección necesaria para llegar a él (dirección ej.: root/sdcard/historias/nombredearchivo).

Luego de identificar el elemento que fue presionado, y de obtener todos los datos necesarios de él, se procede a establecer la conexión con el servidor a través del comando *connect*(“*host*”, “puerto”), este método corresponde a un objeto antes declarado, llamado “conexión”, de tipo FTPClient, al hacer eso el servidor exige una validación de usuario y contraseña. Automáticamente a través del comando *login*(“usuario”, ‘contraseña”) correspondiente al mismo objeto se validan estos datos.

Para lograr cargar el archivo se usó el comando *storefile*(“*name*”, “*stream*”), la variable “*name”* corresponde al nombre recuperado de la etiqueta presionada. Esto indica al “método” que el archivo en el servidor es guardado con ese nombre específico. La variable “*stream”* corresponde al “*InputStream”* del cual puede ser leído el archivo. Así, el “método” entiende de donde extraer el archivo y con qué nombre guardarlo al subirlo a la carpeta compartida en el servidor.

El método *storefile*(“*name*”, “*stream*”) tiene un valor booleano, es por esto que luego de programarlo, se estableció un condicional para asegurar que el archivo sea cargado correctamente. Se usó la herramienta “*Toast”* para mostrar un mensaje de corta duración, indicando al usuario el estado final de la transmisión (carga correcta o carga incorrecta).

Es importante establecer que para la transmisión de los datos se configuró el modo pasivo. De esta manera, el intercambio de datos se hace en puertos predefinidos y seguros.

```
protected void onListItemClick(ListView l, View v, int position, long id) //el metodo responde cuando un elemento de la
{                                                                          //lista es presionado
    File posicion = new File(historias.get(position));                     //se obtiene la posicion del elemento presionado
    nombre = posicion.getName();                                           //se obtiene el nombre del elemento presionado
    sdcard = Environment.getExternalStorageDirectory().toString()+"/Historias";
    path = sdcard + "/" + nombre;                                          //se recupera la ruta del elemento presionado
    conexion = new FTPClient();                                            // se crea un nuevo objeto de tipo "cliente ftp"

    try
    {
        conexion.connect("192.168.1.104", 21);                             //se establece la conexion con el servidor
    }//try
    catch (SocketException e1)
    {
        e1.printStackTrace();
    }//catch
    catch (IOException e1)
    {
        e1.printStackTrace();
    }//catch

    try
    {
        validacion = conexion.login("Prueba", "12345");                    //automaticamente se realiza la validacion de
    }//try                                                                  //nombre de usuario y contrasena
    catch (IOException e1)
    {
        e1.printStackTrace();
    }//catch
```

Ilustración 37: Código Fuente de la Conexión para Cargar un Archivo

Fuente: Elaboración Propia

```
try
{
    conexion.enterLocalPassiveMode();                                      //se establece el tipo de trasmision como pasivo
    conexion.setFileType(FTP.BINARY_FILE_TYPE);                            //se indica que la transmision sera binaria
}//try
catch (IOException e)
{
    e.printStackTrace();
}//catch

try
{
    in = new BufferedInputStream(new FileInputStream (path));              //el inputstream indica desde donde sera
}//try                                                                     //sera leido el documento a cargar
catch (FileNotFoundException e)
{
    e.printStackTrace();
}//catch

try
{
    carga = conexion.storeFile(nombre, in);                                //el archivo es cargado con el nombre indicado
}//try                                                                     //y desde el inputstream indicado
catch (IOException e)
{
    e.printStackTrace();
}//catch io
```

Ilustración 38: Código Fuente para Cargar un Archivo

Fuente: Elaboración Propia

```
if (carga == true)
{
    contexto = getApplicationContext();
    CharSequence text = "Carga correcta";                                  //en caso de no haber error en la transmision
    tiempo = Toast.LENGTH_LONG;
    mensaje = Toast.makeText(contexto, text, tiempo);
    mensaje.show();
}//if carga
else
{
    contexto = getApplicationContext();
    CharSequence text = "Carga incorrecta";                                //en caso de haber error en la transmision
    tiempo = Toast.LENGTH_LONG;
    mensaje = Toast.makeText(contexto, text, tiempo);
    mensaje.show();
}//else carga
```

Ilustración 39: Código Fuente del *Toast* Indicando el Estado de la Transferencia

Fuente: Elaboración Propia

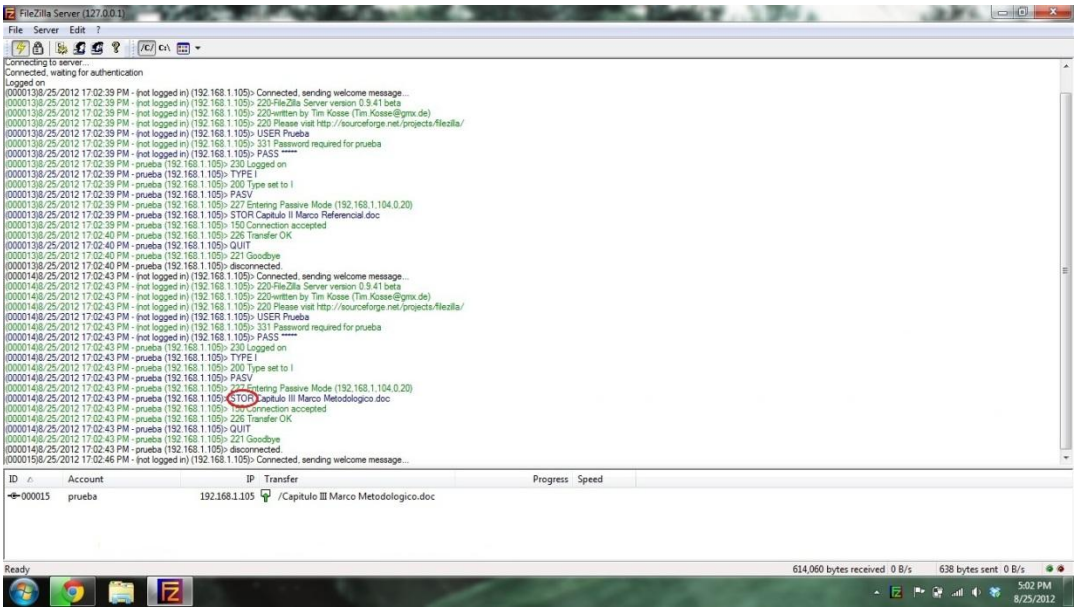

Ilustración 40: Interfaz Gráfica del Servidor Mientras Carga un Archivo

Fuente: Elaboración Propia

Por otra parte, si se presiona la opción Descargar, entonces el *layout* mostrado es uno en el cual se le pide al usuario ingresar su nombre de usuario y su contraseña. Estos dos campos son objetos de tipo "*editText"*. Con objetos de este tipo podemos recuperar el texto ingresado por el usuario.

También en este *layout* hay un objeto de tipo Botón, que se configuró para que cuando sea presionado empiece el proceso de conexión y validación de usuario con el servidor. Esta conexión se establece a través del comando *connect*("*host*", "puerto"). Este "método" corresponde a un objeto antes declarado, llamado "conexión", de tipo FTPClient. Al hacer eso el servidor exige una validación de usuario y contraseña, y usando el comando *login*("usuario", "contraseña") correspondiente al mismo objeto se validan estos datos.

```
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.descargar);

    entrar = (Button)findViewById(R.id.button3);                    //se crea el objeto de tipo boton, que al ser presionado
    entrar.setOnClickListener(new OnClickListener()                 //activara el metodo de conexion con el servidor
    {

        public void onClick(View v)
        {
            usuario = (EditText)findViewById(R.id.editText1);       //se crean objetos de tipo editText de los cuales se
            contrasena = (EditText)findViewById(R.id.editText2);    //recuperaran el nombre de usuario y la contrasena
            conexion = new FTPClient();                             //se crea un nuevo objeto de tipo "cliente ftp"

            try
            {
                conexion.connect("192.168.1.104", 21);              //se establece la conexion con el servidor
            }//try
            catch (SocketException e)
            {
                e.printStackTrace();
            }//catch
            catch (IOException e)
            {
                e.printStackTrace();
            }//catch

            try
            {
                validacion = conexion.login(usuario.getText().toString(), contrasena.getText().toString());
            }//try                    //se hace en login con el nombre de ususario y la contrasena recuperada de los editText
            catch (IOException e)
            {
                e.printStackTrace();
            }//catch
```

Ilustración 41: Código Fuente de la Conexión para Descargar un Archivo

Fuente: Elaboración Propia

El "método" *login* tiene un valor booleano (*boolean*), con lo cual, a través de un condicional se puede verificar que la validación de nombre de usuario y contraseña ha sido correcta. De ser así, a través de un *Intent* se llama a la librería ZXING, y se instancia a la aplicación *Barcode Scanner* que permite leer y procesar el código QR. En cambio, si la validación no ha sido correcta, a través de un "*Toast"* se le indica al usuario que no ha sido posible validar los datos y se le invita a intentarlo de nuevo.

```
if (validacion == true)
{
    lectorQR = new Intent("com.google.zxing.client.android.SCAN"); //se instancia la libreria ZXING de google
    lectorQR.putExtra("SCAN_MODE", "QR_CODE_MODE" ); //para leer el codigo QR, de ella se espera recuperar el
    startActivityForResult(lectorQR, 0); //estado del escaneo y que tipo de codigo QR es el escanedo
}//if validacion
else
{
    contexto = getApplicationContext();
    CharSequence text = "Nombre de usuario o contrasena no validos, por favor vuelva a intentar";
    tiempo = Toast.LENGTH_LONG;  //en caso de haber un error en la validacion de usuario
    mensaje = Toast.makeText(contexto, text, tiempo);
    mensaje.show();
}//else
```

Ilustración 42: Código Fuente de la Instancia y *Toast* de Error

Fuente: Elaboración Propia

Si la lectura QR se realiza con éxito, entonces el "método" resultado empieza a procesar la información recolectada. La lectura (código QR), se recupera en formato *String* para facilitar su manejo. También, se declara la ruta de memoria externa del equipo, ya que es ahí donde se quiere guardar el archivo descargado.

Para asegurar una transferencia confiable se activó el modo pasivo en la transmisión y se configuró la comunicación para que la transferencia de datos sea binaria. Esto provee un manejo más efectivo de los datos compartidos.

Para descargar el archivo se usó el comando *retrievefile*("código", "*stream*"), la variable "código" corresponde al nombre del archivo que se quiere descargar. En este caso, esta variable toma el valor recuperado de la lectura QR. La variable "*stream"* corresponde al "*OutputStream"* en el cual puede ser escrito el archivo. Así, el método entiende que archivo debe descargar, y donde debe guardarlo.

El método *retrievefile(*"código", "*stream*") tiene un valor booleano. Es por esto que luego de programarlo se establece un condicional para asegurar que el archivo fue descargado correctamente. Se usa la herramienta "*Toast"* para mostrar un mensaje, indicando al usuario el estado final de la transmisión (descarga correcta o descarga incorrecta).

```
public void onActivityResult(int requestCode, int resultCode, Intent lectorQR) //el metodo se activa al obtener una lectura QR
{
    if (requestCode == 0)
    {
        if (resultCode == RESULT_OK)
        {
            codigo = lectorQR.getStringExtra("SCAN_RESULT");      //el codigo se recupera en forma de string
            sdcard = Environment.getExternalStorageDirectory().toString()+"/Historias"; //ruta de la sdcard
            modo = FTPClient.PASSIVE_LOCAL_DATA_CONNECTION_MODE;  //se establece una conexion de tipo pasiva con el servidor

                try
                {
                    conexion.enterLocalPassiveMode();
                    conexion.setFileType(FTP.BINARY_FILE_TYPE);  //se establece una transmision binaria
                }//try
                catch (IOException e)
                {
                    e.printStackTrace();
                }//catch io

                try
                {
                    out = new BufferedOutputStream(new FileOutputStream (sdcard+"/"+codigo)); //se indica el outputstream
                }//try                                                  //desde donde podra ser recuperado el archivo a descargar
                catch (FileNotFoundException e)
                {
                    e.printStackTrace();
                }//catch filenotfound

                try
                {
                    descarga = conexion.retrieveFile(codigo, out);  //se descarga el archivo con el nombre que indique el
                }//try                                               //codigo QR y desde el outputstream indicado
                catch (IOException e)
                {
                    e.printStackTrace();
                }//catch io
```

Ilustración 43: Código Fuente para Descargar un Archivo

Fuente: Elaboración Propia

```
if (descarga == true)
{
    contexto = getApplicationContext();
    CharSequence text = "Descarga correcta";                  //en caso de no haber error en la transmision
    tiempo = Toast.LENGTH_LONG;
    mensaje = Toast.makeText(contexto, text, tiempo);
    mensaje.show();
}//if descarga
else
{
    contexto = getApplicationContext();
    CharSequence text = "Descarga incorrecta";               //en caso de haber error en la transmision
    tiempo = Toast.LENGTH_LONG;
    mensaje = Toast.makeText(contexto, text, tiempo);
    mensaje.show();
}//else descarga
```

Ilustración 44: Código Fuente del *Toast* Indicando el Estado de la Transferencia

Fuente: Elaboración Propia

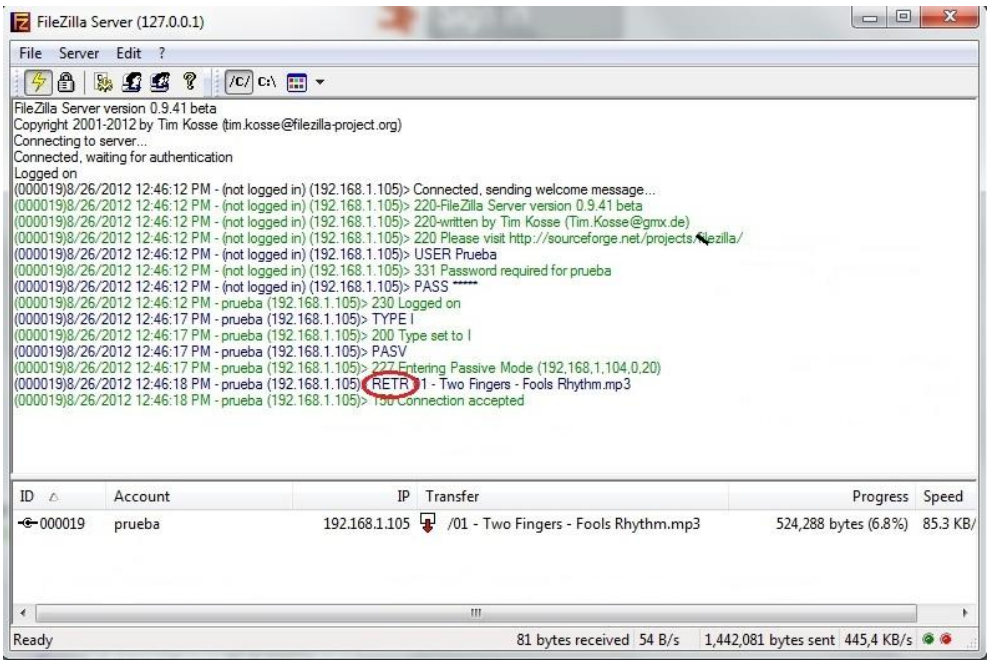

Ilustración 45: Interfaz Gráfica del Servidor Mientras Descarga un Archivo
Fuente: Elaboración Propia

En una aplicación Android es indispensable el manejo de permisos para poder usar recursos de hardware y software del equipo. Es por eso que el *Android Manifest* tiene vital importancia en el desarrollo de la aplicación.

En el *Android Manifest*, se establecieron los permisos necesarios para usar los recursos en nuestra aplicación: Internet, WiFi, cámara y el derecho a escribir la memoria externa.

También se le indicó a la aplicación que se instancian otras clases y librerías dentro de ella.

A continuación se muestra la configuración completa del *Android Manifest* de la aplicación desarrollada.

```
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="example.ftp"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />
        <uses-permission android:name="android.permission.INTERNET" />
        <uses-permission android:name="android.permission.WIFI" />
        <uses-permission android:name="android.permission.CAMERA" />
        <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".FTP"
            android:label="@string/title_activity_ftp" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
            <action android:name="com.google.zxing.client.android.SCAN" />
            <category android:name="android.intent.category.DEFAULT" />
            <activity android:name="Cargar"></activity>
            <activity android:name="Descargar"></activity>
    </application>
</manifest>
```

Ilustración 46: *Android Manifest* de la Aplicación

Fuente: Elaboración Propia

4.IV Prueba Piloto

Con el fin de comprobar el funcionamiento de la aplicación en un ambiente de trabajo se procedió a probar la aplicación y el servidor en la red de una clínica. La clínica en donde se realizaron las pruebas fue la Sanatrix en la Urbanización Campo Alegre.

En esta prueba se procuró buscar la facilidad de la configuración y adaptación de la red para que el servidor FTP pudiese ser instalado y fuese capaz de transferir las historias médicas propias de la institución.

Para poder adaptar la Red de la Clínica para el correcto uso del servidor FTP, fue necesario hacer el *forwarding* de los puertos 20 y 21 hacia la dirección IP del servidor. En esta prueba el servidor FTP se instaló en nuestra computadora personal.

El último paso para comenzar con las pruebas de la transferencia fue la configuración de la dirección del servidor FTP en la aplicación.

Por razones de seguridad de la clínica, solo se pudo hacer las pruebas a través de la red WiFi y no de la 3G. La clínica alegó que el uso de la red 3G expondría sus historias a la red pública, a pesar de la previa explicación de las características de seguridad del sistema.

Para verificar el funcionamiento del proyecto en redes 3G se hicieron pruebas en nuestros respectivos hogares. En esta prueba se transfirieron las historias médicas en ambas modalidades (3G y WiFi) a fin de comparar el rendimiento de ambas. Se utilizaron las redes 3G de Movistar y de Digitel.

Capitulo 5. Resultados

Basado en los objetivos específicos propuestos al empezar el desarrollo de esta investigación, se presentan a continuación los resultados del Trabajo Especial de Grado, con los cuales se busca hacer una recopilación de los productos alcanzados en su desarrollo.

5.I Digitalización de Documentos.

Al recopilar información de archivos digitales de historias médicas en centros de salud del Distrito Capital, nos dimos cuenta de que no existe una única manera de digitalizar y presentar documentos.

Algunas de las instituciones conocidas en la ciudad de Caracas, ya cuentan con pequeños directorios de historias médicas digitalizas.

Existen médicos que por iniciativa propia han empezado a digitalizar documentos para uso particular en las consultas; facilitando así, el manejo de información al momento de actualizar las historias.

Es necesario darle versatilidad al sistema dejando abierta la posibilidad de que cada institución pueda escoger su manera de tener digitalizadas las historias médicas.

Para logar que el rango de alternativas de digitalización de las historias médicas sea lo más amplio posible, se probó la transmisión de distintos formatos (.txt, .pdf, .jpg, .doc, .xls, .png, .rar, etc.), desde el servidor hasta el dispositivo móvil. De los formatos probados, se llega a la conclusión de que los que mejor funcionan y los que ofrecen mayor facilidad de manejo en equipos móviles son los archivos con formato: .pdf, .doc, .xls y .jpg.

Hay que tomar en cuenta que para la visualización de documentos con estos formatos, a excepción de archivos .pdf, no es necesaria una aplicación adicional en el teléfono, ya que el sistema operativo **Android**, ofrece el visor y editor de documentos de office *DocumentsToGo*, y un visor de imágenes, que es compatible con la mayoría de formatos existentes en el mercado. Para el caso específico de documentos .pdf, es necesario descargar (del mercado de aplicaciones oficial *PLAY STORE*) la aplicación gratuita y oficial, ADOBE READER, visor ampliamente conocido y confiable para archivos con extensión .pdf.

5.II Determinación de Especificaciones de Código QR a utilizar.

Una de las principales ventajas que ofrecen los códigos **QR**, es su fácil generación. Estos pueden ser generados desde páginas Web o desde programas gratuitos descargados a la computadora.

Para tener mayor robustez en caso de una falla en la red y para poder tener más opciones a la hora de crear los códigos **QR** se decidió descargar e instalar en la computadora del Servidor el programa *Quantum QR Generator*. Este programa permite al usuario crear los códigos **QR** de hasta 40 tamaños diferentes y con los 4 niveles de corrección.

Para garantizar que cada código **QR** sea único, cada uno de éstos, guarda el número de cédula del paciente. Estos códigos se imprimen y se guardan en la habitación o sala donde se encuentra el paciente. De esta manera queda accesible para el uso del médico.

Como la cantidad de caracteres que tiene cada código es reducida, se podrían usar códigos **QR** de tamaño 1. Sin embargo para dar más redundancia ante posibles

errores se usa de tamaño 3. El cual, con un nivel de corrección H (*High*), permite almacenar 58 caracteres numéricos y 35 alfanuméricos.

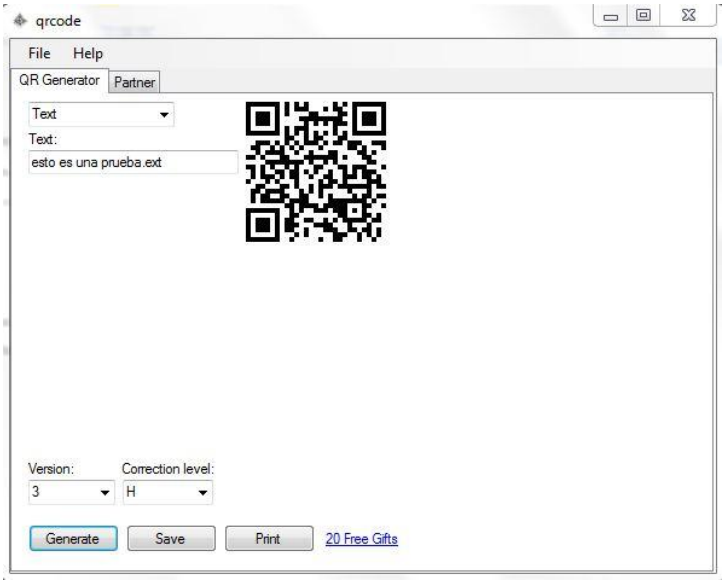

Ilustración 47: Código QR de Prueba

Fuente: Elaboración Propia

Es importante tener en cuenta que el código QR consta de dos partes. La parte principal, que representa la identificación de paciente (numero único, y nombre del archivo en el servidor), y la extensión que corresponde al tipo de archivo solicitado. Esta extensión queda de parte de la clínica y su forma de manejar los documentos. De obviar esta extensión, la aplicación Android es incapaz de identificar al documento en la base de datos del Servidor.

5.III Instalación y conectividad de servidor FTP.

Para la implementación del servidor se tomaron en cuenta todas las características de la estructura, para hacer un uso óptimo de ella.

En principio se discutió el sistema operativo sobre el cual se alojaría el servidor. Tomando en cuenta la gran oferta de interfaces disponibles, se evaluó la capacidad que tenía cada una para ofrecer un manejo confiable, fácil acceso capacidad de mantenimiento y asistencia remota.

Luego de estudiar sistemas operativos como, **Windows** (cualquier versión), **Ubuntu** (cualquier versión), **Linux** (cualquier versión), **Debian** (cualquier versión), **Solaris.** Se llegó a la conclusión de que el que cumplía con todas las características deseadas era **Windows**.

Se justifica el uso de **Windows** como sistema operativo (SO de ahora en adelante) para el Servidor porque: a pesar de ser un SO pesado, con interfaz gráfica no necesariamente indispensable, siempre fue prioritario tomar en cuenta donde se encontraría y quien operaría el servidor. Recordemos que la máquina central fue propuesta para estar instalada en la sala de emergencias de una clínica, a la cual solo tendrán acceso las personas pertenecientes a la institución que estén autorizadas. De esta forma queremos asegurar que en caso de fallas pueda haber una persona que sin profundos conocimientos de sistemas operativos sea capaz de dar una primera atención al servidor, mientras se establece una conexión de asistencia remota.

Una vez decidido el SO, se decidió usar el programa *FileZilla Server*, como software servidor, esta decisión se basa en la confiabilidad que genera el programa y el prestigio de la organización que lo gestiona (***The FileZilla Project***). Además, esta solución puede ser descargada oficial y gratuitamente desde la página [16].

Dentro del grupo de características que ofrece la configuración del servidor, está la creación de usuarios y grupos de usuarios con carpetas compartidas bajo ciertas condiciones. En el caso del sistema diseñado, se crearon nombres de usuario para cada médico que labora en la institución, específicamente en el área de emergencias, para la cual se creó un grupo llamado Emergencias. Cada usuario cuenta con una

clave única configurada por él, que le da acceso al servidor y a la carpeta compartida, que, según sus privilegios es capaz de ver.

Los privilegios otorgados por grupo dependen de la necesidad que tiene cada uno de ellos al momento de consultar un archivo paciente. El *FileZilla* cuenta con una opción para permitir a cada usuario ver, escribir, borrar y guardar archivos en la carpeta asociada a él. De esta forma, si se otorga una "súper cuenta" (médicos), se puede tener acceso total a los datos del paciente, y si se tiene una cuenta limitada (enfermería, farmacia, etc.), se puede visualizar solo la información que a ese departamento compete. Gracias a esto, se puede asegurar la privacidad de la historia médica y su correcto uso dentro de la institución.

También, el software *FileZilla* permite dar más seguridad a la comunicación a través del protocolo **TLS** (***Transport Layer Security***) y/o su antecesor **SSl** (***Secure Sockets Layer***). Con estos protocolos el servidor genera certificados para la autenticación de la información que entra y sale de él. De esta forma, independientemente de la red sobre la que se esté trabajando (WiFi o UMTS), la información va cifrada, y solo las partes involucradas en la comunicación son capaces de traducir esa información para poder ser leída.

El servidor *FileZilla* ofrece la configuración de modo de operación. Hay que recordar que el servidor FTP puede funcionar en modo activo o pasivo. En nuestro, caso se configura el FileZilla en su modo pasivo, en el cual, luego de recibir el **ACK** *(Acknowledgment*) del servidor, el cliente le indica por cual puerto empezará la transmisión de los datos.

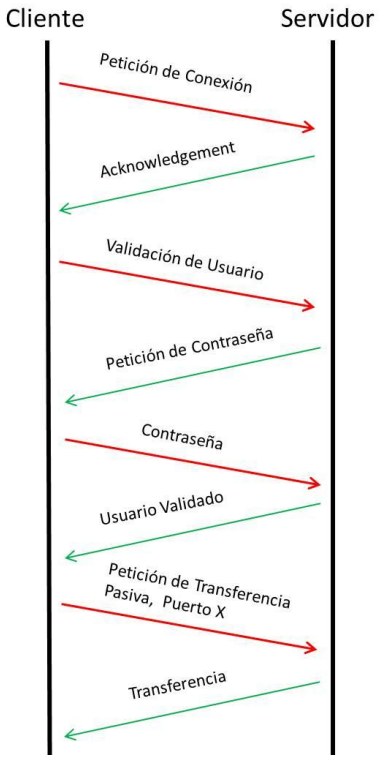

Ilustración 48: Fases de Conexión con el Servidor *FileZilla*

Fuente: Elaboración Propia

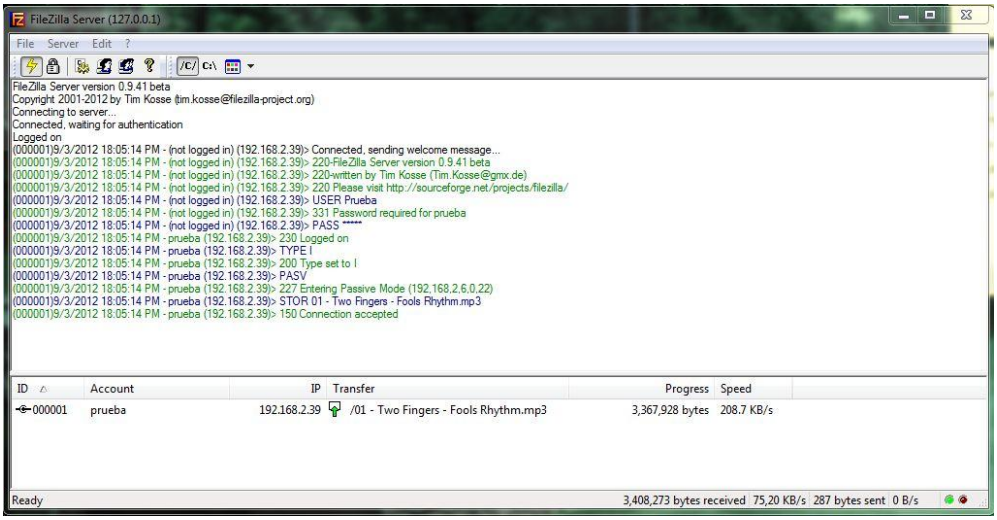

Ilustración 49: Interfaz del Servidor *FileZilla* en Transferencia

Fuente: Elaboración Propia

5.IV Historia Médica

Al no haber un formato predeterminado para el manejo de historias médicas, el sistema se adapta a cualquier tipo de formato. En este caso en específico, al formato prestado por la Clínica Sanatrix.

Dentro de los documentos manejados se encuentran imágenes en formato .jpg, resultantes de estudios de radiología y tomografía. También se encuentran resultados de laboratorio que son procesados en formato .pdf.

Adicionalmente, se maneja un documento en formato .xls (Excel), para tener información concerniente a los datos personales del paciente.

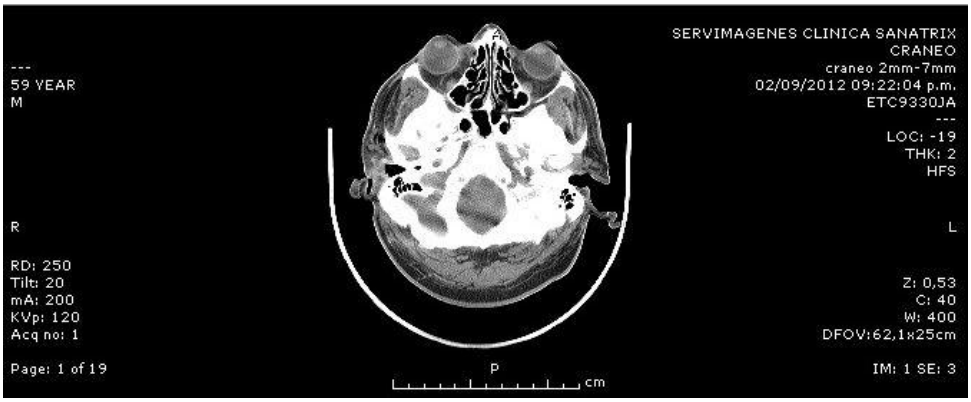

Ilustración 50: Tomografía Digital de Cráneo

Fuente: Clínica Sanatrix

5.V Funcionamiento de aplicación en Android.

La aplicación Android tiene un funcionamiento muy básico; tomando en cuenta su función se busca que el manejo sea muy sencillo y que la aplicación se presente de forma agradable al momento de leer el código QR y al momento de elegir el documento que se desea cargar.

El primer paso es indicar a la aplicación qué acción se quiere llevar a cabo: cargar un archivo o descargar un archivo.

Ilustración 51: *Layout* Menú de Opciones

Fuente: Elaboración Propia

En caso de querer cargar un archivo, inmediatamente se despliega una lista que contiene a todos los elementos que están guardados en la carpeta Historias de la memoria externa *SDCard*. Al presionar cualquiera de estos elementos el documento correspondiente sube inmediatamente al servidor, creando un nuevo archivo o sobrescribiendo un documento con el mismo nombre.

Ilustración 52: *Layout* Lista de Archivos en la *SD Card*

Fuente: Elaboración Propia

En ambos casos puede existir una descarga correcta o incorrecta que se le indica al usuario a través de un mensaje en pantalla.

En caso de elegir la opción descargar, se solicita al médico su nombre de usuario y su contraseña para poder acceder al servidor. De ser inválidos estos datos, se despliega un mensaje sugiriendo al usuario volver a ingresar la información.

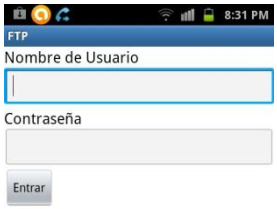

Ilustración 53: *Layout* Ingreso de Datos de Usuario y Contraseña

Fuente: Elaboración Propia

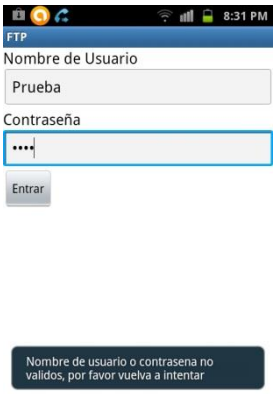

Ilustración 54: Mensaje de Error en Validación de Nombre de Usuario o Contraseña

Fuente: Elaboración Propia

De ser correcta la validación, inmediatamente, se muestra en la pantalla el barrido de imagen correspondiente a la cámara; el médico debe presentar el código QR correspondiente al documento que quiera descargar.

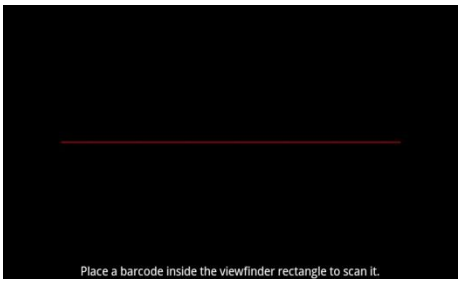

Ilustración 55: *Layout* Lector QR

Fuente: Elaboración Propia

En ambos casos puede existir una descarga correcta o incorrecta que se le indica al usuario a través de un mensaje en pantalla.

5.VI Prueba Piloto

5.VI.I Prueba Piloto Clínica

A continuación se presentan los resultados de las pruebas utilizando la red WiFi de la clínica:

Wifi Clínica				
Documento	Tamaño	Extensión	Carga	Descarga
1. craneo2	39.5KB	.jpg	Sí	Sí
2. torax1	33.2KB	.jpg	Sí	Sí
3. resultados de Laboratorio	157KB	.pdf	Sí	Sí

Tabla 2: Datos de la Prueba con WiFi en la Clínica

Fuente: Elaboración Propia

Para respaldar los resultados obtenidos en esta prueba se hizo un video donde se graba el proceso de descargar y cargar las historias médicas.

5.VI.II Prueba Hogar

5.VI.II.I Red WiFi

A continuación se presentan los resultados de las pruebas utilizando la red WiFi:

Wifi Hogar						
Documento	Tamaño	Extensión	Carga	Tiempo Carga	Descarga	Tiempo Descarga
1. craneo2	39.5KB	.jpg	Sí	1s	Sí	5s
2. torax1	33.2KB	.jpg	Sí	1s	Sí	4s
3. resultados de Laboratorio	157KB	.pdf	Sí	1s	Sí	5s
4. 12345678	145KB	.rar	Sí	1s	Sí	7s

Tabla 3: Datos de la Prueba con WiFi en el Hogar

Fuente: Elaboración Propia

A continuación se presentan las imágenes de los procesos de carga y descarga de cada uno de los documentos:

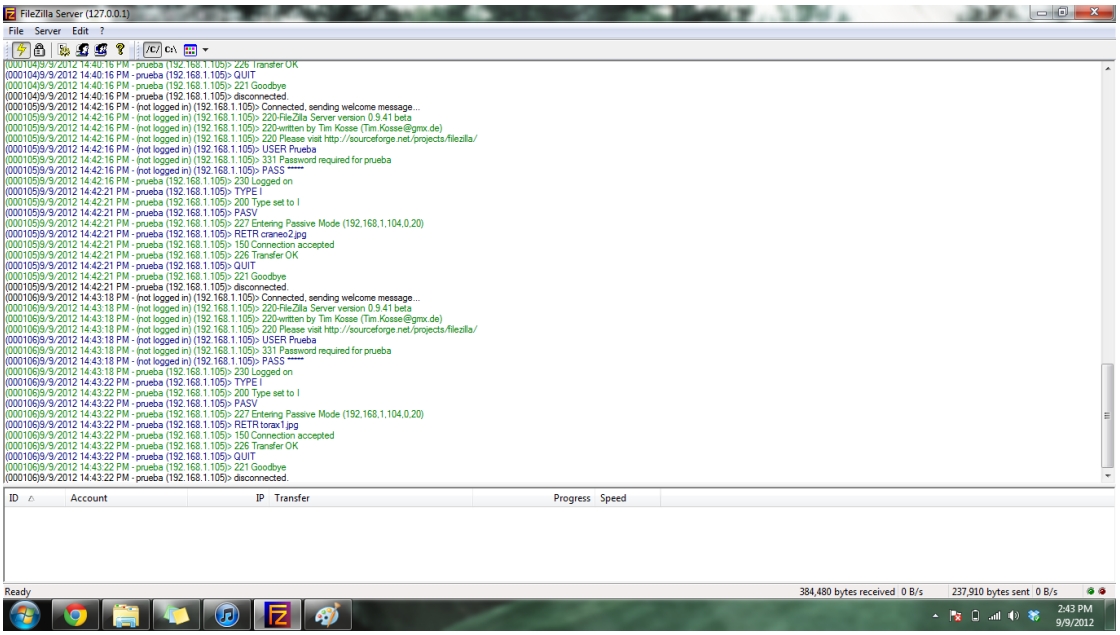

Ilustración 56: Interfaz del Servidor Descargando el Archivo **Torax1.jpg**

Fuente: Elaboración Propia

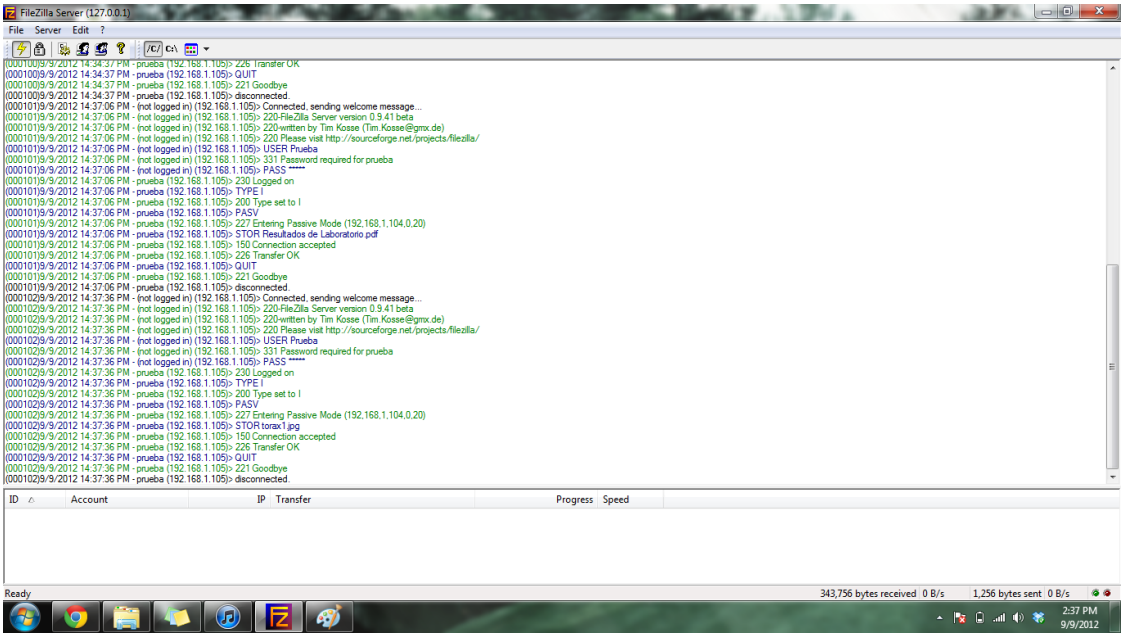

Ilustración 57: Interfaz del Servidor Cargando el Archivo **Torax1.jpg**

Fuente: Elaboración Propia

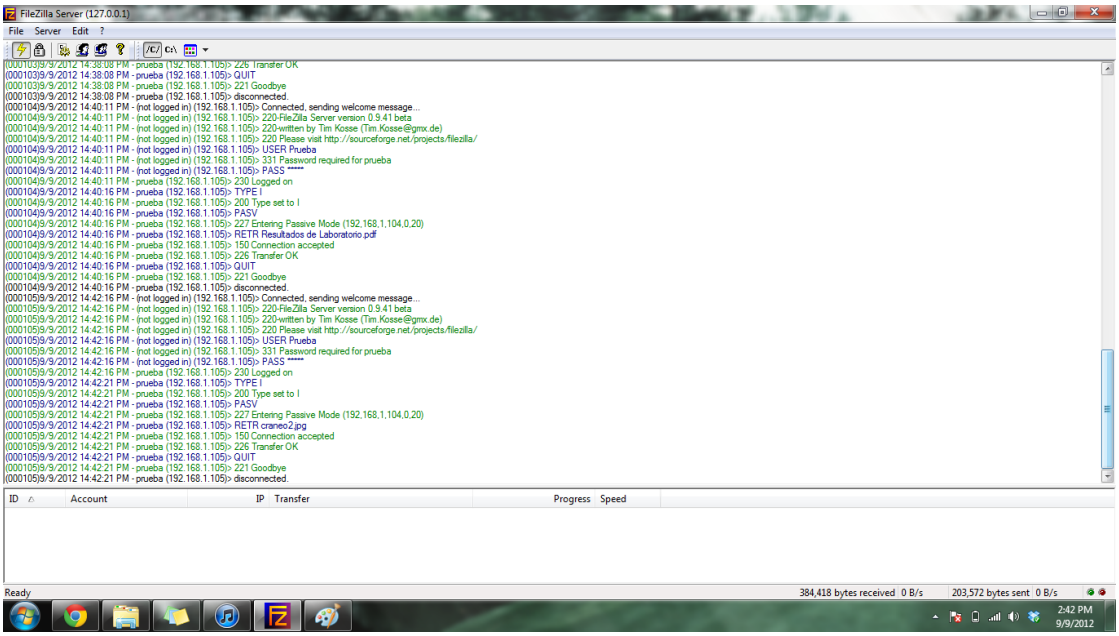

Ilustración 58: Interfaz del Servidor Descargando el Archivo **Craneo2.jpg**

Fuente: Elaboración Propia

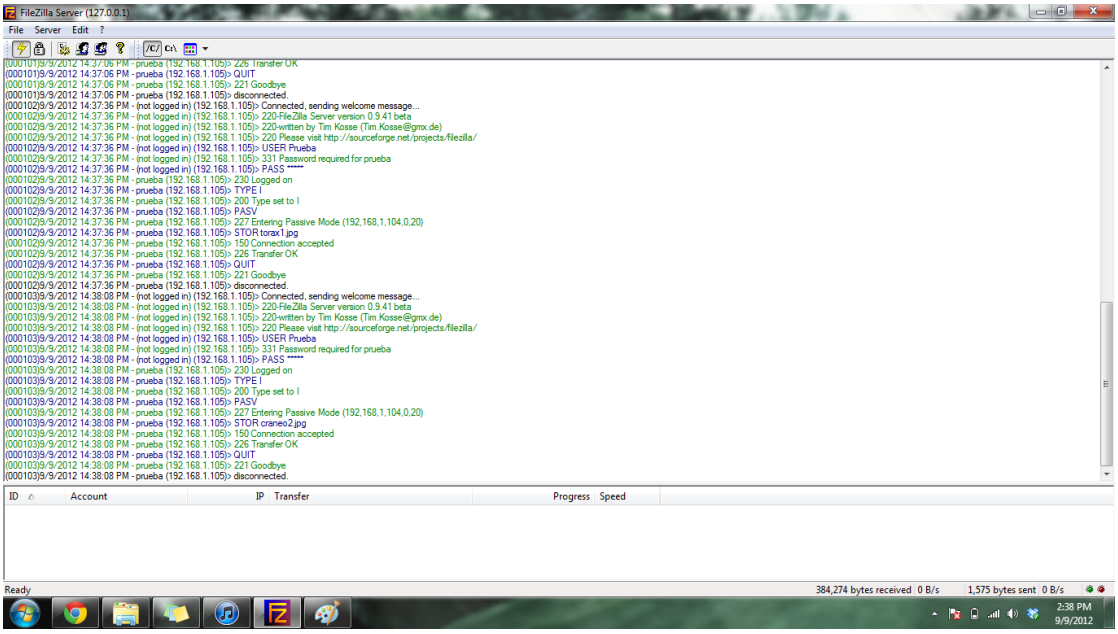

Ilustración 59: Interfaz del Servidor Cargando el Archivo **Craneo2.jpg**

Fuente: Elaboración Propia

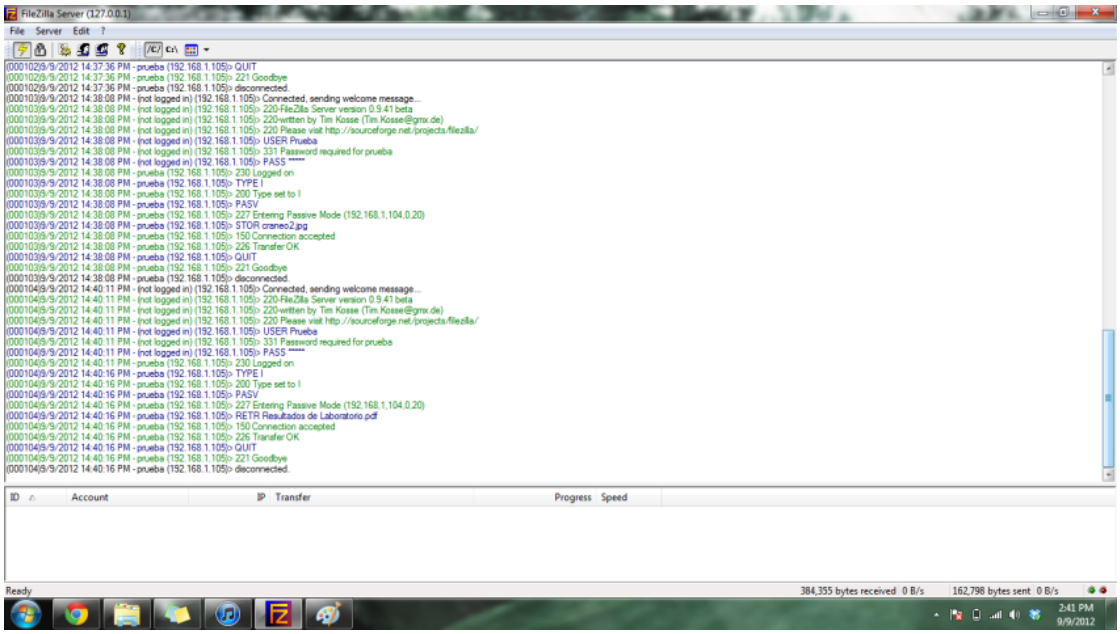

Ilustración 60: Interfaz del Servidor Descargando el Archivo **Resultados.pdf**

Fuente: Elaboración Propia

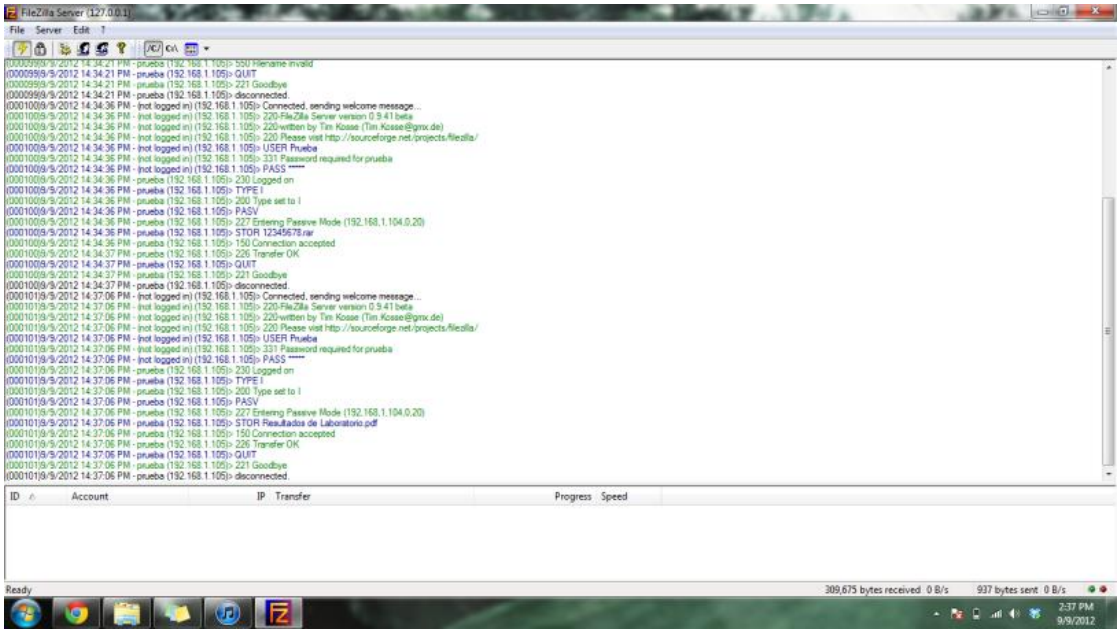

Ilustración 61: Interfaz del Servidor Cargando el Archivo **Resultados.pdf**

Fuente: Elaboración Propia

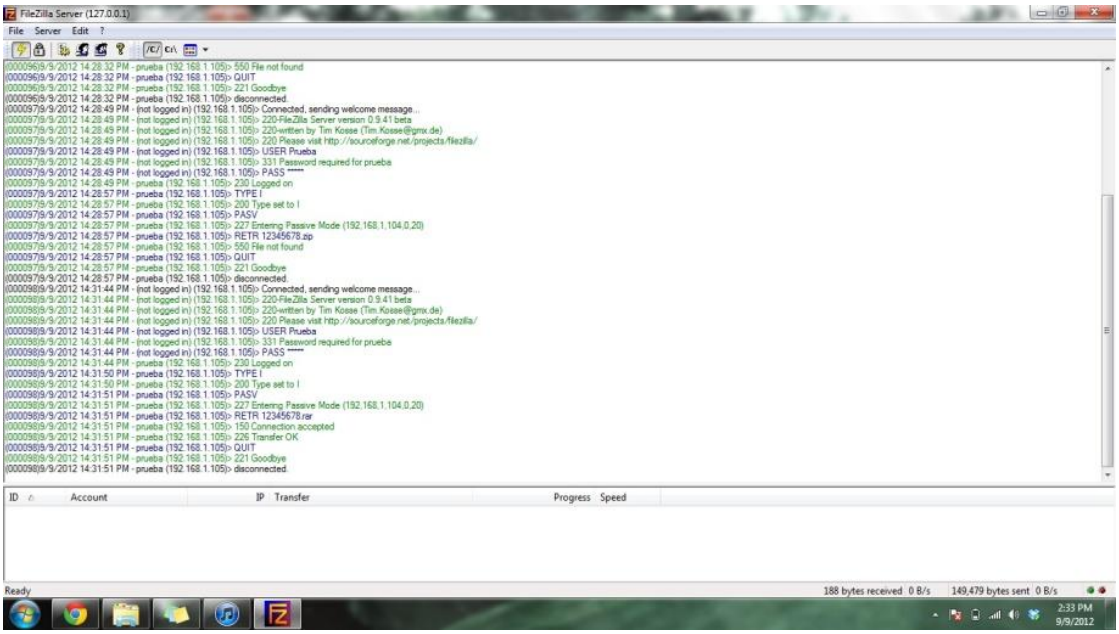

Ilustración 62: Interfaz del Servidor Descargando el Archivo **1234567.rar**

Fuente: Elaboración Propia

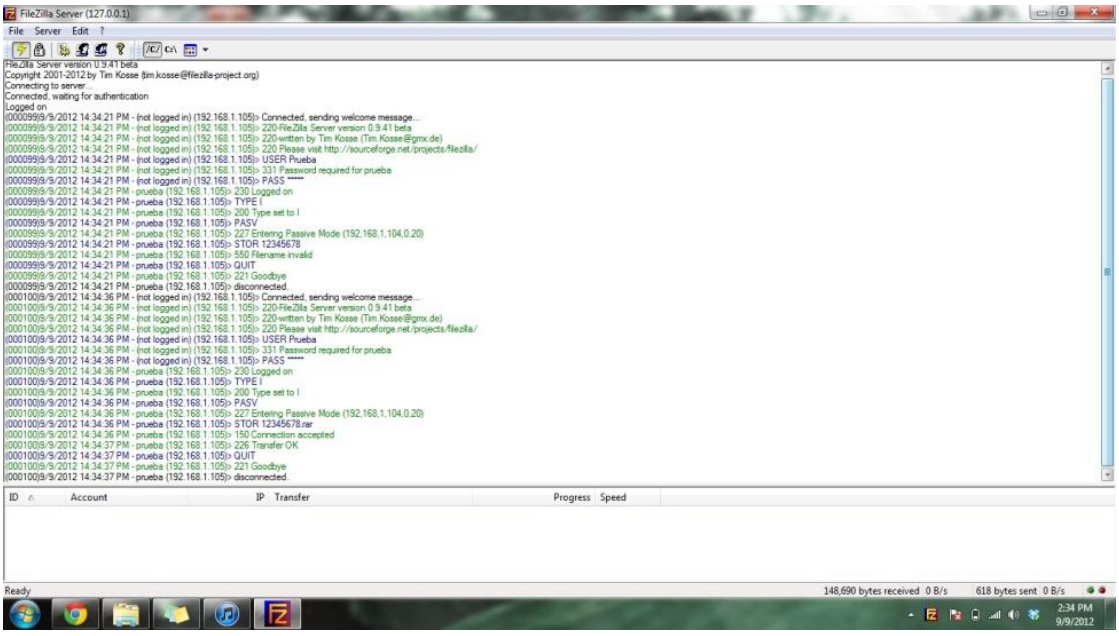

Ilustración 63: Interfaz del Servidor Descargando el Archivo **1234567.rar**

Fuente: Elaboración Propia

5.VI.II.II Red 3G Movistar

A continuación se presentan los resultados de las pruebas utilizando la red 3G de Movistar:

3G Movistar						
Documento	Tamaño	Extensión	Carga	Tiempo Carga	Descarga	Tiempo Descarga
1. craneo2	39.5KB	.jpg	Sí	3s	Sí	47s
2. torax1	33.2KB	.jpg	Sí	4s	Sí	15s
3. resultados de laboratorio	157KB	.pdf	Sí	2s	Sí	18s
4. 12345678	145KB	.rar	Sí	2s	Sí	34s

Tabla 4: Datos de la Prueba con Red 3G Movistar en el Hogar

Fuente: Elaboración Propia

A continuación se presentan las imágenes de los procesos de carga y descarga de cada uno de los documentos:

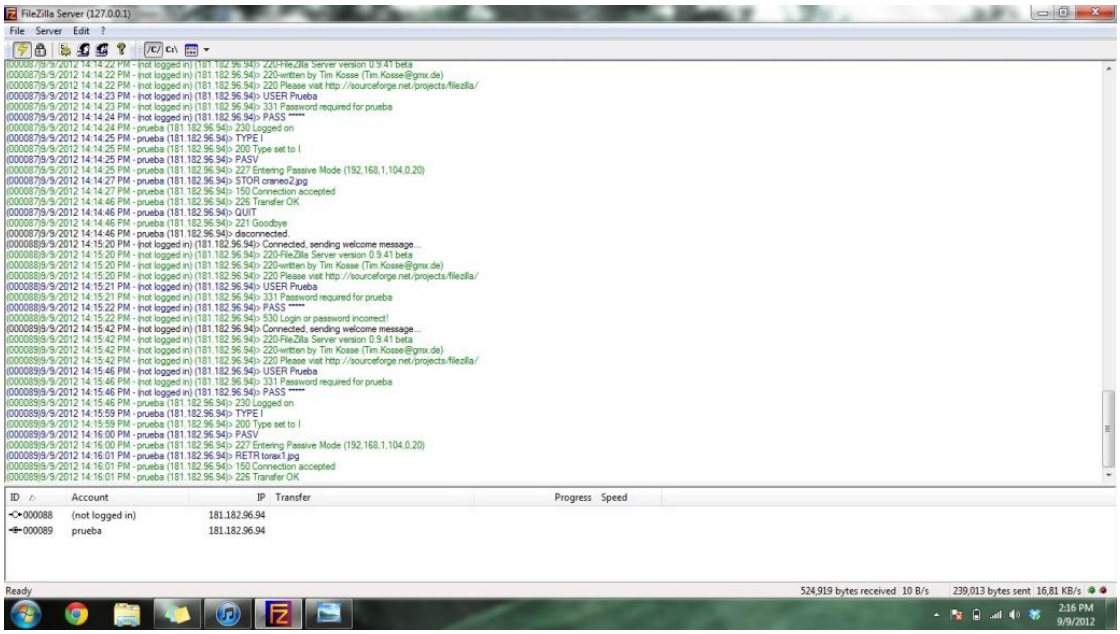

Ilustración 64: Interfaz del Servidor Descargando el Archivo **Torax1.jpg**

Fuente: Elaboración Propia

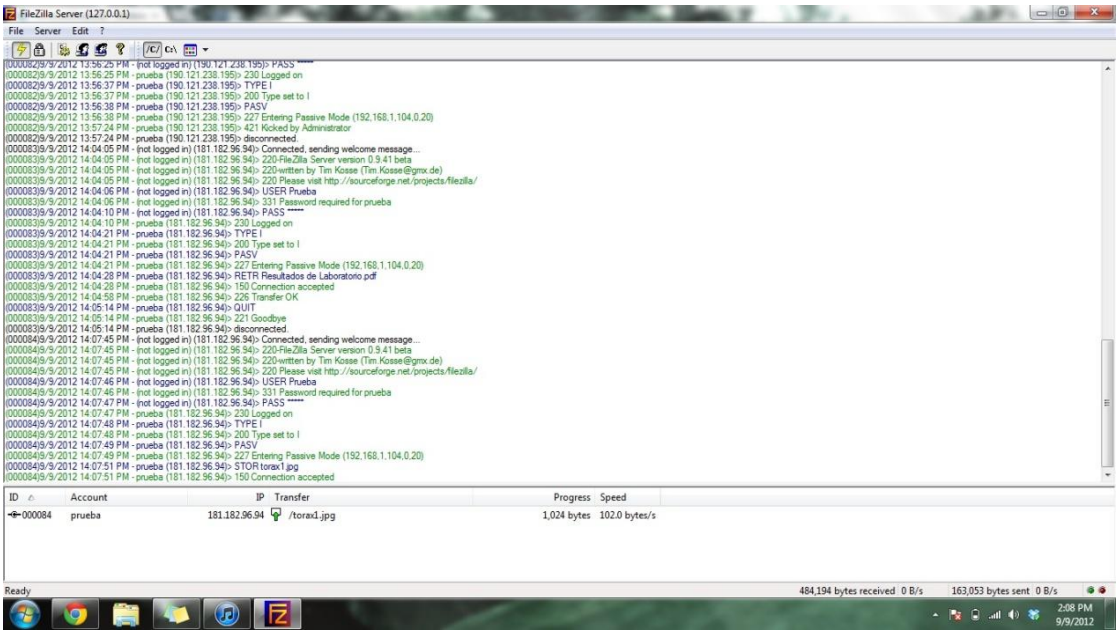

Ilustración 65: Interfaz del Servidor Cargando el Archivo **Torax1.jpg**

Fuente: Elaboración Propia

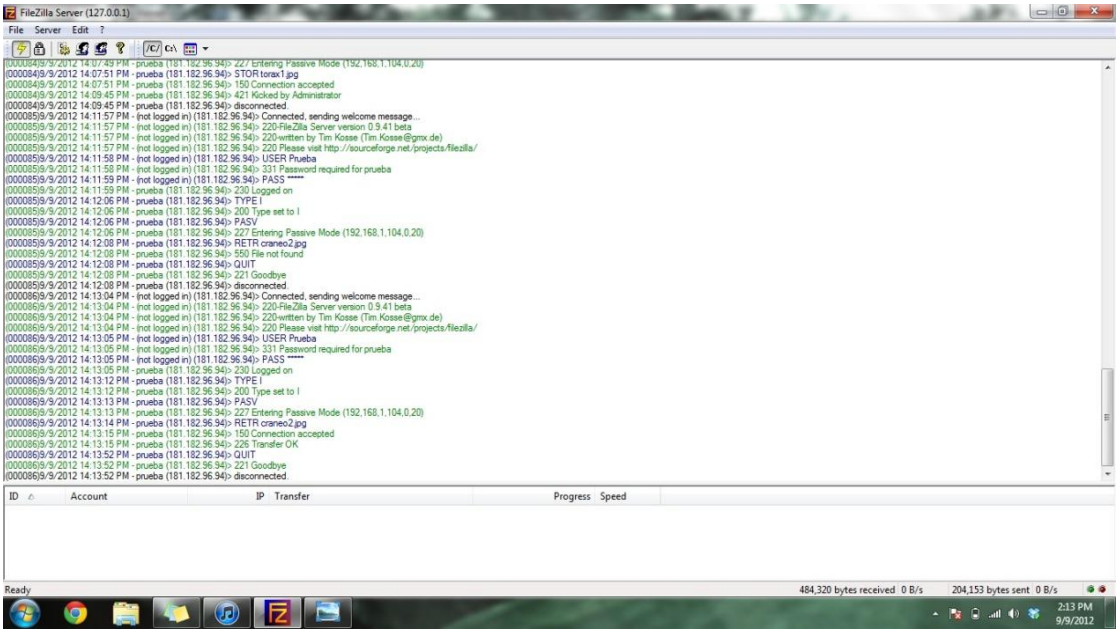

Ilustración 66: Interfaz del Servidor Descargando el Archivo **Craneo2.jpg**

Fuente: Elaboración Propia

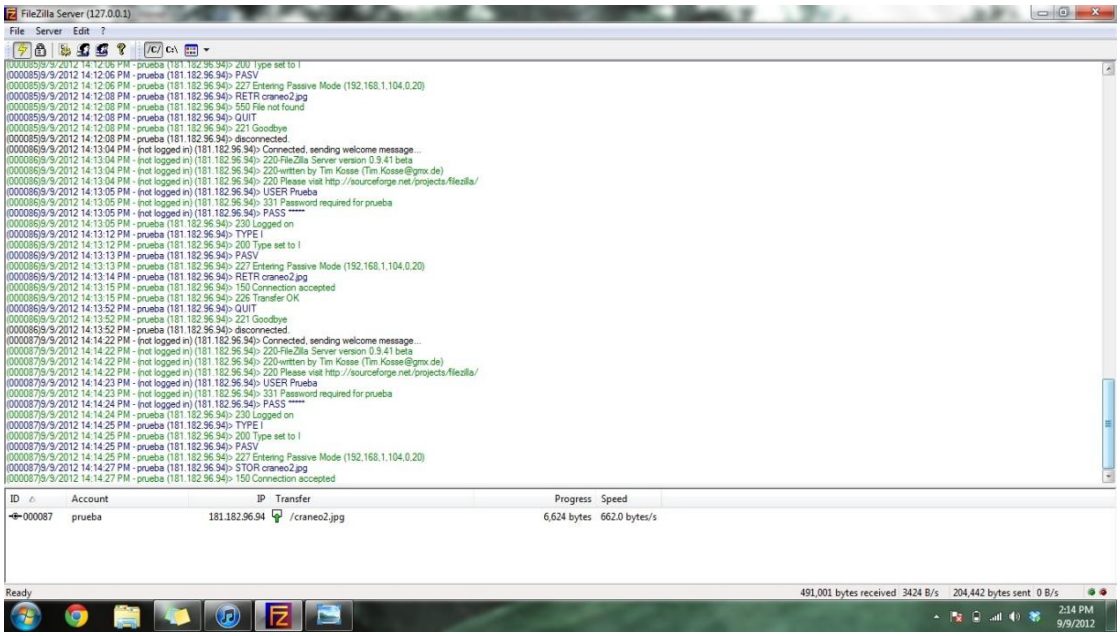

Ilustración 67: Interfaz del Servidor Cargando el Archivo **Craneo2.jpg**

Fuente: Elaboración Propia

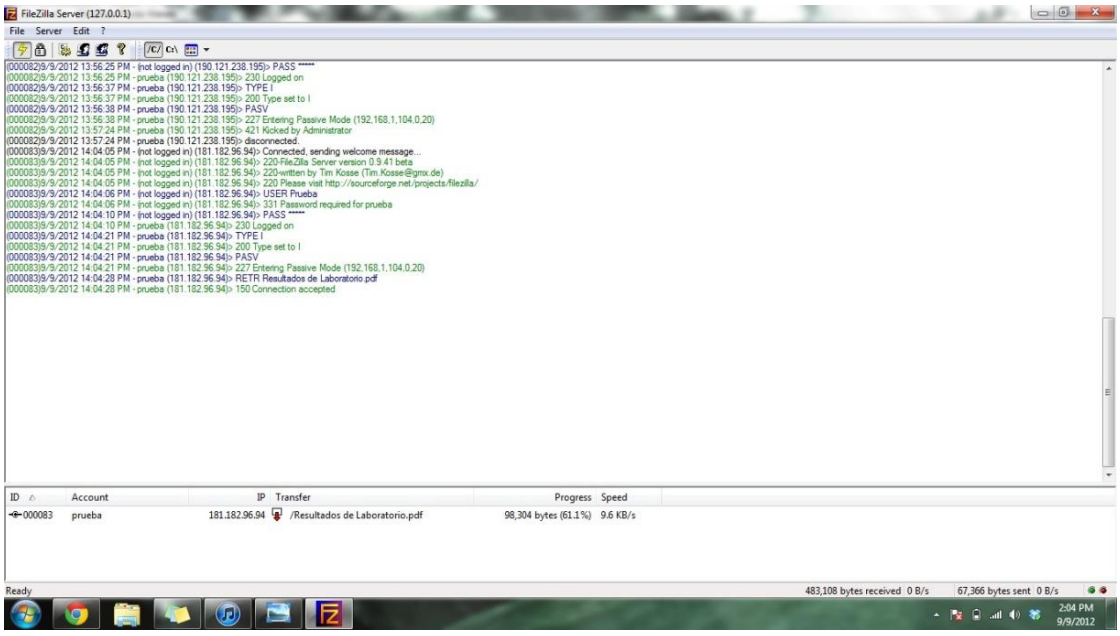

Ilustración 68: Interfaz de Servidor Descargando el Archivo **Resultados.pdf**

Fuente: Elaboración Propia

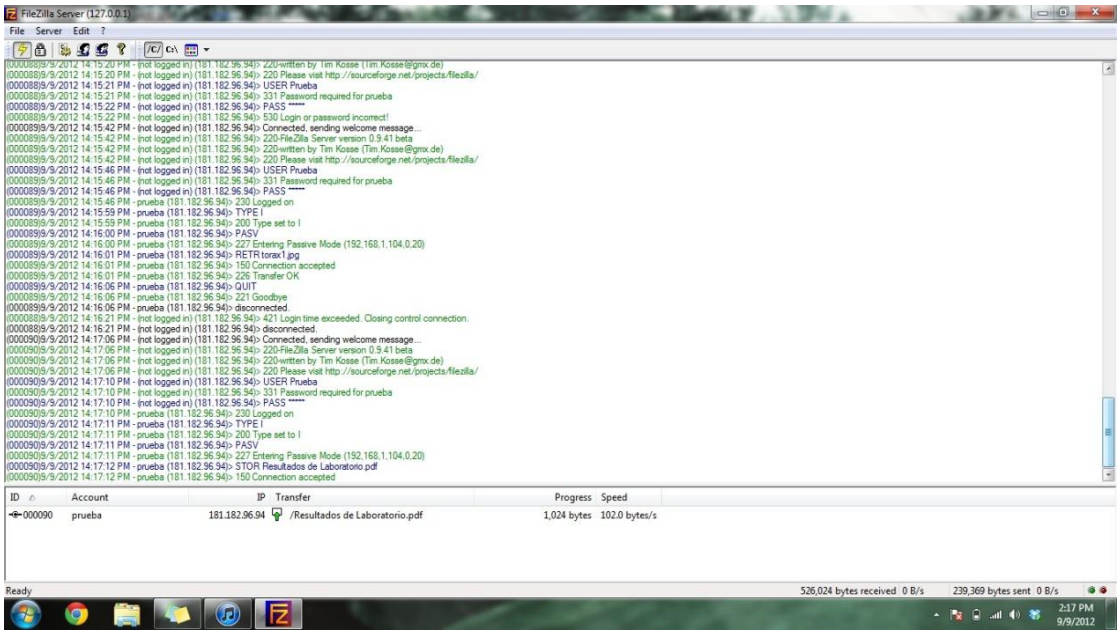

Ilustración 69: Interfaz de Servidor Cargando el Archivo **Resultados.pdf**

Fuente: Elaboración Propia

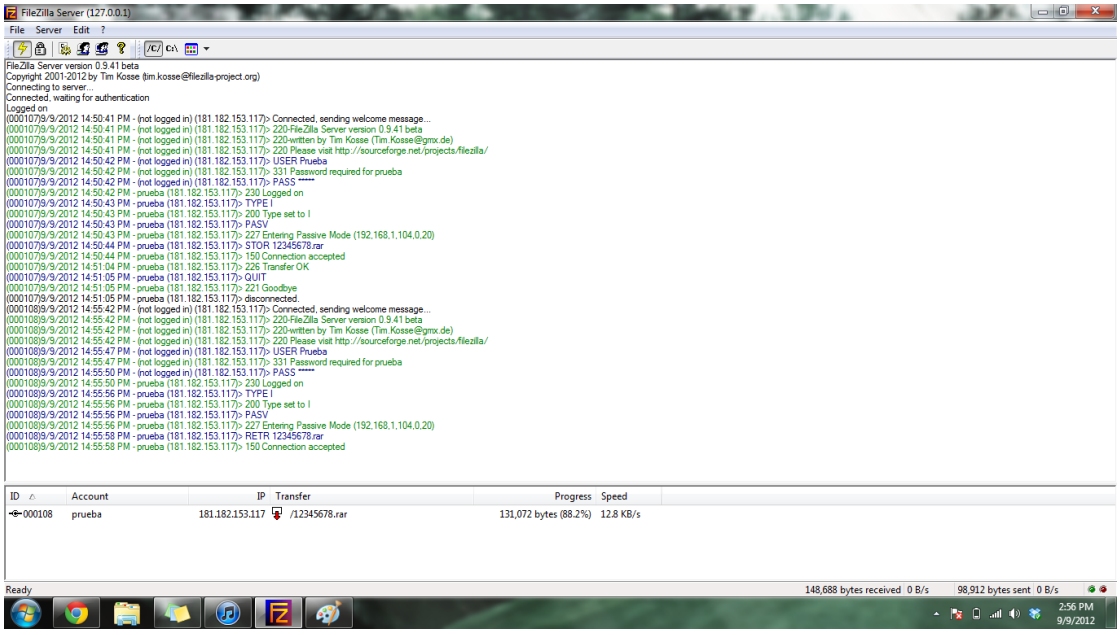

Ilustración 70: Interfaz del Servidor Descargando el Archivo **123456.rar**

Fuente: Elaboración Propia

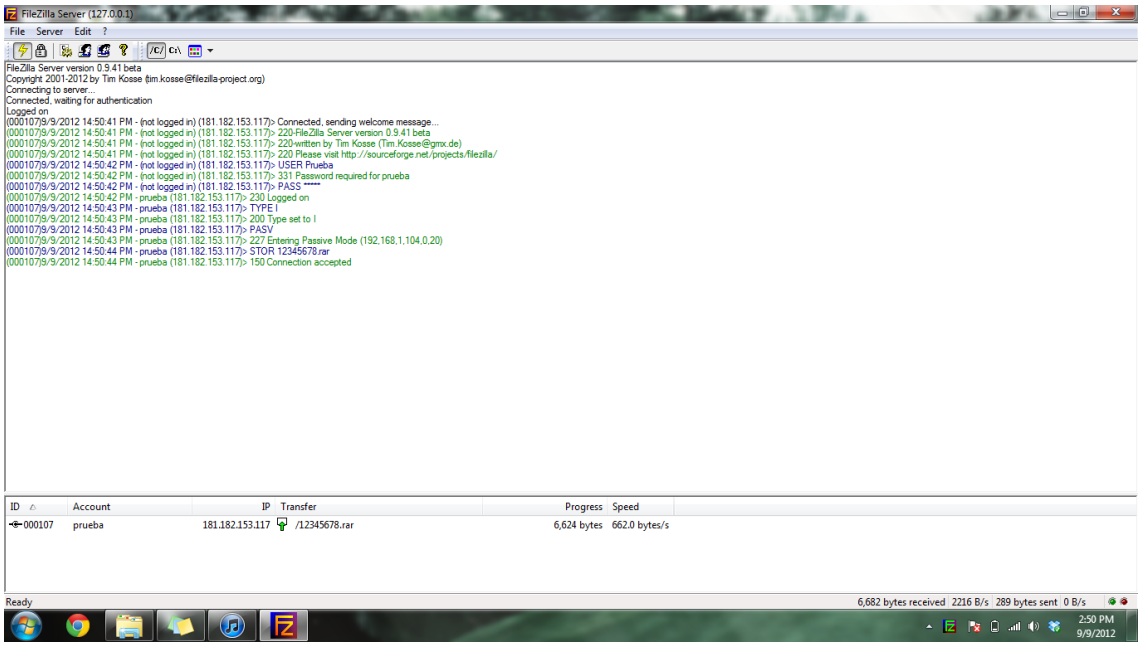

Ilustración 71: Interfaz del Servidor Cargando el Archivo **123456.rar**

Fuente: Elaboración Propia

5.VI.II.III Red 3G Digitel

A continuación se presentan los resultados de las pruebas utilizando la red 3G de Digitel:

3G Digitel				
Documento	Tamaño	Extensión	Carga	Descarga
1. craneo2	39.5KB	.jpg	No	No
2. torax1	33.2KB	.jpg	No	No
3. resultados de Laboratorio	157KB	.pdf	No	No
4. 12345678	145KB	.rar	No	No

Ilustración 72: Datos de la Prueba con Red 3G Digitel en el Hogar

Fuente: Elaboración Propia

Los archivos no pudieron ser transferidos a través de la red 3G de Digitel.

Una vez hechas las pruebas se demostró que la manera más rápida, segura y eficiente de usar esta aplicación es a través de las redes WiFi. A través de este tipo de redes hubo un 100% de efectividad en la transferencia de las historias médicas y tiempos 3 veces menores que los de las redes 3G.

5.VII Estimado de Costo

En este caso, se consideran costos de implementación al dinero necesario para iniciar con el sistema, esta inversión se hace una sola vez.

Los costos de operación son los costos recurrentes, aquellos que se presentarán mensualmente y que son necesarios para el correcto funcionamiento del sistema.

Costos de Implementación	Equipo Móvil con Android OS(celular o tableta)	Bs.: 8.500,00
	Computador con Almacenamiento Sugerido de 1 TB	Bs.: 10.000,00
	Router Inalambrico (diseñado para areas amplias)	Bs.: 600,00
Costo total de Implementación		Bs.: 19.100,00
Costos de Operación	Plan de Datos Móviles (Movistar, Digitel, Movilnet)	Bs: 300,00
	Servicio de Internet (Domestico)	Bs.: 400,00
Costo Total de Operación (mensual)		Bs.: 400,00

Tabla 5: Estimado de Costos Totales

Fuente: Elaboración Propia

Capitulo 6. Conclusiones y Recomendaciones.

6.I Conclusiones

Con este trabajo de grado se logró elaborar un sistema en el que a través de la identificación de pacientes con códigos QR se puede visualizar y editar su historia médica almacenada en un servidor FTP, utilizando una aplicación Android para dispositivos móviles. Esta transferencia se logra de forma segura e inmediata, y es posible realizarla por medio de redes WiFi y redes 3G.

La aplicación en el dispositivo móvil se creó utilizando el desarrollador Android. Al ser código abierto existe un gran número de posibilidades para programar y acceder a documentación sobre este sistema operativo, lo que permite mayor facilidad a la hora de realizar cualquier cambio o mejora que se quiera implementar.

Para almacenar las historias médicas se utilizó un servidor FTP que permite la creación de usuarios con distintos niveles de acceso y permisos. Estos servidores también cuentan con sistemas de cifrado y claves que otorgan seguridad en la transferencia de archivos. Para cumplir con estos requisitos, de manera sencilla y económica, se utilizó *FileZilla*.

Se comprobó el funcionamiento de la aplicación y el servidor al realizar varias pruebas tanto en redes de clínicas como en otras localidades. Se identificó que el funcionamiento en redes WiFi es más efectivo que en redes 3G. En las primeras, se logró la transferencia exitosa de las historias en el 100% de las ocasiones y en tiempos menores a 2 segundos. Este hecho implica que la recepción de las historias médicas desde el momento que el especialista lee el código QR al que lo recibe fue casi inmediato. En las redes 3G, la transferencia fue más lenta, en el caso de Movistar la transferencia se logró en un 100% con tiempo promedio de 10 segundos y en caso de DIGITEL no fue posible realizar la transferencia.

Al utilizar programas de códigos abiertos y gratuitos la instalación de la aplicación en una red médica resulta bastante económica y sencilla por lo que el campo de esta aplicación es muy amplio. Estas facilidades sumadas al hecho de que el sistema puede transferir varios tipos de archivos, permite que el proyecto funcione tanto en consultorios como en salas de emergencias. Estas ventajas, sus requerimientos económicos y simplicidad demuestran que para resolver algunos problemas no hacen falta costosas y complejas soluciones sino más bien estudiar minuciosamente la forma y los recursos con que resolverlos.

6.II Recomendaciones

El objetivo principal de este trabajo de grado fue crear una aplicación que pudiese obtener de forma inmediata una historia médica almacenada en un servidor, para así poder disminuir sustancialmente el tiempo de atención a los pacientes de una clínica y hospital, tanto en consulta como en emergencia.

Esta aplicación fue hecha exclusivamente para Android, por lo que se recomienda para futuros proyectos adaptarla para otros sistemas operativos (Blackberry, Apple y Windows Mobile), con el fin de expandir el campo de uso de ésta.

También como proyecto futuro, se recomienda la creación de una identificación única por ciudadano, que identifique su historial médico y que pueda utilizar en cualquier institución médica del país. Para que este proyecto funcione es necesario que exista un estándar único de almacenamiento de historias. En donde si bien el paciente puede tener historias en distintas clínicas u hospitales, éstas están almacenadas bajo la misma identificación.

Para blindar aún más el funcionamiento de la aplicación Andorid se sugiere modificar los códigos para ampliar los niveles de seguridad y control.

Bibliografía

[1] I. System Storage, «Soluciones de Almacenamiento de Gama Media Preparadas para Cloud,» IBM, [En línea]. Available: http://www-03.ibm.com/systems/storage. [Último acceso: Diciembre 2011].

[2] Apple Inc., «iCloud,» Apple Inc., [En línea]. Available: http://www.apple.com/icloud. [Último acceso: Diciembre 2011].

[3] B. Oposicionestic, «Oposiciones Tic,» 08 Junio 2011. [En línea]. Available: http://oposicionestic.blogspot.com/2011/06/arquitectura-cliente-servidor.html. [Último acceso: Diciembre 2011].

[4] P. Renaud, Introduction to Client/Server Systems: A Practical Guide for Systems Professionals, New York, US: John Wiley & Sons, 1996.

[5] A. Tanenbaum, Sistemas Distribuidos: Principios y Paradigmas, Ciudad de Mexico, Mexico: Pearson Education, 1997.

[6] B. Forouzan, Transmision de Datos y Redes de Comunicaciones, Madrid, Espana: McGraw Hill, 2001.

[7] Net Storming, «FTP Activo vs. FTP Pasivo,» 22 Junio 2009. [En línea]. Available: http://www.netstorming.com.ar/2009/06/22/ftp-activo-vs-pasivo. [Último acceso: Febrero 2012].

[8] IETF. [En línea]. Available: tools.ietf.org/html/rfc5246. [Último acceso: Febrero 2012].

[9] Microsoft, «Microsoft Technet,» [En línea]. Available: technet.microsoft.com/en-us/library/cc784450(v=ws.10).aspx. [Último acceso: Febrero 2012].

[10] Denso Wave, «About QR Code,» Denso Wave, [En línea]. Available: http://www.denso-wave.com/qrcode/index-e.html. [Último acceso: Diciembre 2011].

QR Planet, «Generador QR,» [En línea]. Available:

11] http://www.qrplanet.com/es/generador-qr-code. [Último acceso: Diciembre 2011].

12] Engineers Garage, «What is Android?: Introduction,» [En línea]. Available: http://www.engineersgarage.com/articles/what-is-android-introduction?page=2. [Último acceso: Enero 2012].

13] A. Developers, «What is Android?,» [En línea]. Available: http://developers.android.com/guide/what-is-android.html. [Último acceso: Enero 2012].

14] «Android Everywhere,» [En línea]. Available: http://www.android.com/developers/. [Último acceso: Enero 2012].

15] CNet, «Quantum QR Code Generator,» [En línea]. Available: http://download.cnet.com/quantum-qr-generator/3000-2650_4-75542736.html. [Último acceso: Marzo 2012].

16] «FileZilla Project,» [En línea]. Available: http://filezilla-project.org/download.php?type=server.

17] Tecnologia Hecha Palabra, «Soluciones & Tecnologia: Como Utilizar un Servidor FTP,» [En línea]. Available: http://www.tecnologiahechapalabra.com/datos/soluciones/implementacion/articulo.asp?i1520. [Último acceso: Diciembre 2011].

18] Mas Adelante, «Que es un Servidor FTP?,» [En línea]. Available: http://www.masadelante.com/faqs/servidor. [Último acceso: Diciembre 2011].