UNIVERSIDAD SAN FRANCISCO DE QUITO

Colegio de Ciencias e Ingeniería

Diseño de Prototipo Doméstico de Video Vigilancia con Cámaras IP por internet

Franklin Marcelo Barreno Masabanda

Fausto Pasmay, MBA., Director de Tesis

Tesis de grado presentada como requisito para la obtención del título de Ingeniero de Sistemas

Quito, Mayo 2013

Universidad San Francisco de Quito

Colegio de Ciencias e Ingeniería

HOJA DE APROBACIÓN DE TESIS

Diseño de Prototipo Doméstico de Video Vigilancia con Cámaras IP por internet

Franklin Marcelo Barreno Masabanda

Fausto Pasmay, MBA

Director de Tesis …….

Luis Miguel Prócel, MSc

Miembro del Comité de Tesis …….

Ximena Córdova, PHD

Decana de la Escuela de Ingeniería

Colegio Politécnico …….

Quito, Mayo 2013

Firma:

Nombre: Franklin Marcelo Barreno Masabanda

C. I.: 1802709483

Fecha: Mayo 2013

Dedicatoria

Dedico este trabajo con mucho cariño a mi familia, y mis seres queridos que han tenido que dejar este mundo, de manera especial a mi madre quien día a día, con su afecto, cariño y comprensión me apoyó en forma incondicional, para quien guardo sentimientos de infinito amor.

Agradecimientos

Mis más sinceros agradecimientos a la Universidad San Francisco de Quito, por haberme permitido hacer realidad uno de mis más preciados sueños, alcanzar el título de Ingeniero, de manera particular a todos y cada uno de mis maestros, quienes con su sabiduría alimentaron cada día, cada mes y cada año los conocimientos que tengo con respecto a las ciencias de la ingeniería, así como a la noble Institución Policial del Ecuador por la oportunidad.

A Fausto Pasmay, y mis profesores quienes sabiamente me han sabido dirigir en la realización de este trabajo y por su valiosa colaboración. Un agradecimiento a mis compañeros policías y de la Universidad por su apoyo brindando y en especial a Robert Balarezo.

Resumen

Un sistema de video vigilancia con cámaras IP por internet, es una estructura de dispositivos tecnológicos como cámaras IP, switchs, routers, que permiten captar y transmitir señales de video y de audio a través de una red a otros dispositivos de red como un PC o dispositivos móviles. La arquitectura básica para un sistema de video vigilancia por internet, son las cámaras IP que captan imágenes y estas son enviadas a través de un navegador web, por un servidor web, en la que los usuarios están en la posibilidad de visualizar, y gestionar el video ya sea de forma local o remota en tiempo real a través del navegador web.

Para el presente proyecto se busca la implementación de un prototipo de video vigilancia adaptada para las necesidades del usuario, en donde, los costos y la implementación resulten factibles, para la mayoría de las personas. Permitiendo también mejorar la percepción de seguridad de las personas al contar con elementos tecnológicos para contribuir a la seguridad y preservación de sus bienes. Por último permite a las personas optimizar sus recursos que poseen dentro de sus domicilios como son el servicio de internet y dispositivos móviles y no invertir contingentes cantidades de dinero para adquirir sistemas de seguridad con video cámaras a empresas en el mercado nacional e internacional.

Abstract

A video surveillance system with IP cameras on internet is a structure of technological devices such as IP cameras, switches, routers, that can capture and transmit video and audio via a network to other network devices such as a PC or mobile devices. The basic architecture for a video surveillance system on the Internet, are IP cameras that capture images and these are sent through a web browser by a web server in which users are in the ability to view, and manage video either locally or remotely in real time through a web browser.

For this project seeks the implementation of a prototype video surveillance adapted to the needs of a home, where, the implementation and costs to be accessible for most people. Allowing also improve the perceived safety of the people, and to contribute to the safety and preservation of their property. Finally allows people to optimize their resources they have in their homes such as the Internet and mobile devices and not invest contingent amounts of money to buy security systems with video cameras to businesses in domestic and international markets.

TABLA DE CONTENIDOS

LISTA DE ILUSTRACIONES

LISTA DE TABLAS

1. INTRODUCCION

1.1. Antecedentes

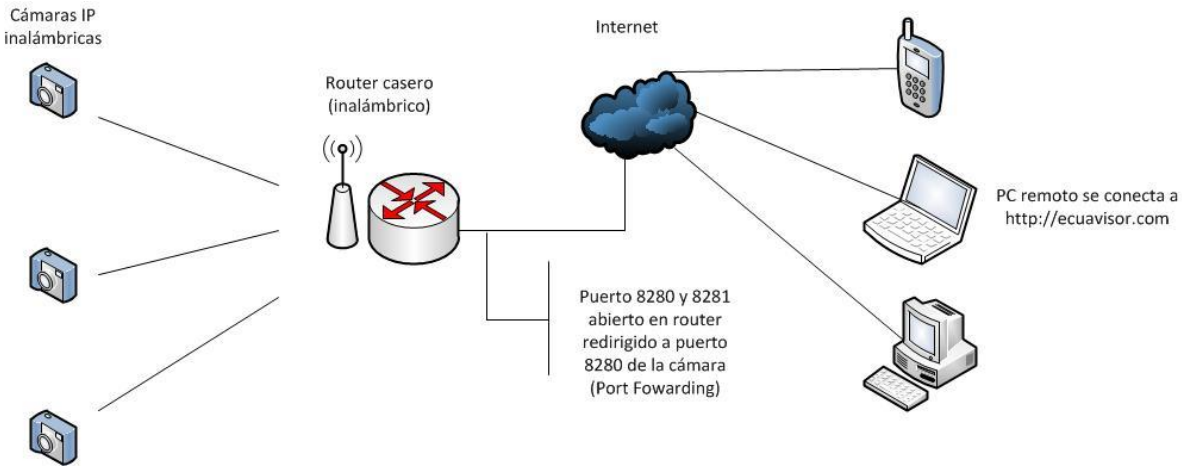
Una cámara de video por Internet es un dispositivo que capta y trasmite una señal de video y audio digital a través de una red IP a otros dispositivos de red como puede ser una PC o un teléfono inteligente. Mediante la dirección IP, un servidor web, y protocolos de streaming de video (server push), los usuarios están en la posibilidad de visualizar, y gestionar el video ya sea de forma local o remota en tiempo real por medio del uso de navegadores web. (García, 2004).

El sistema de video vigilancia se inició para cubrir las necesidades del sector privado reduciendo así los costos de sus operaciones por perdidas externas e internas(bienes muebles e inmuebles), entre otros factores, pero últimamente se ha incrementado en el sector público constituyéndose en política de seguridad ciudadana para la prevención del cometimiento del delito.

Las políticas de seguridad ciudadana se hacen cada vez más complejas debido a las nuevas formas de violencia y de criminalidad que van apareciendo, por lo que se hace necesario el desarrollo de nuevas tecnologías sofisticadas y de punta que permitan poner al servicio de la colectividad dichos avances.

La arquitectura básica que requiere un sistema de vigilancia de video por medio de cámaras IP está constituido por las cámaras de red IP que deben cumplir con los requisitos para la red, los servidores de almacenamiento de video o computadoras, el software que permite la gestión de los videos, acceso al internet, y elementos que componen una red de datos que son: cables (de

corriente eléctrica y de red), routers, switchs, o sistemas inalámbricos, según se representa en la IlustraciónN°1.



Fuente: Generación propia

Ilustración N° 1. Esquema básico de un sistema de video vigilancia IP.

Para el presente proyecto se utiliza las cámaras IP, un router inalámbrico, un servidor de TOMCAT y la aplicación web alojada en el dominio ecuavisor.com con IP pública (200.125.156.101), que se considera como el software de gestión de las cámaras IP. (García, 2004).

En la actualidad, no existen sistemas de video vigilancia de bajo costo, al contrario algunas empresas extranjeras y ecuatorianas que ofrecen el sistema de cámaras IP de vigilancia en sus diferentes modelos son de costos elevados, sistemas de almacenamiento conocidos como DVR, NVR que por sus siglas Digital Video Record, y Network Video Record respectivamente, que se utilizan como almacenamiento de video también tienen costo adicional, las empresas ofrecen además software de gestión de video, y en definitiva accesorios del CCTV (Closed Circuit Televisión).

El CCTV por sus siglas significa Circuito Cerrado de Televisión, es un sistema que permite el control y registro de las actividades de un espacio físico a través de imágenes capturadas por las cámaras y que tiene un acceso limitado y de cierta manera restringida al contenido de las imágenes en cuanto a la resolución. Los componentes de un sistema de CCTV básico se compone de varios componentes en relación con el sistema de video vigilancia por cámaras IP. Este último tiene menos elementos, pero una de las principales diferencias radica que el CCTV no permite acceso remoto desde el internet con la misma funcionalidad y facilidad que los sistemas de vigilancia con cámaras IP por Internet.

Es decir contratar servicios de seguridad a través de video vigilancia resultan costosos ya que su implementación en ciertos casos son complejas. A nivel ecuatoriano existen empresas que proveen el servicio pero que están enfocados en el ámbito empresarial e industrial, más no al doméstico o masivo. [1]

Con la implementación del Sistema Doméstico de Video Vigilancia con cámaras IP por internet va a beneficiar a las personas que dispongan del servicio de internet, ya que permite monitorear remotamente sectores considerados por el usuario como críticos en los lugares de residencia, preservando de esta manera sus bienes y colaborando con la prevención del cometimiento del delito en sectores aledaños. Además las personas optimizan el uso del internet que poseen en sus domicilios al dedicarle a la trasmisión de video cuando se encuentran ausentes de sus domicilios.

[1] Video Vigilancia: CCTV usando vídeos IP, nuevas tecnologías pág. 13

1.2. Planteamiento del Problema

Para lo cual se formula y se delimita el problema planteado en el presente proyecto de la siguiente manera:

1.2.1 Formulación

Diseñar un Sistema Doméstico de Video Vigilancia con Cámaras IP por internet orientado al monitoreo remoto.

1.2.2 Delimitación

La implementación del sistema de video vigilancia por internet se realizará en un domicilio con dos cámaras IP, un router, y acceso a internet con un ancho de banda suficiente para la trasmisión de video. Luego el usuario estará en la capacidad de acceder a las dos cámaras IP colocadas en lugares considerados como críticos en el domicilio, desde un sitio fuera del mismo a través de un navegador web, el mismo podrá interactuar con las cámaras IP realizando el monitoreo del domicilio.

El software de gestión de video ofrece la posibilidad de interactuar con las cámaras, logrando movimiento en un rango de 270 grados, así como permite la configuración de eventos con audio y movimientos de objetos. El software que utiliza el prototipo de video vigilancia es una aplicación web desarrollado con tecnología JAVA que actuará como un servidor web, desde el cual se puede controlar al resto de cámaras.

Para que el servidor web sea visible desde internet, básicamente es necesario:

- Contratar un dominio y hosting con soporte TOMCAT.

- Usar port-forwarding en el router para que las conexiones externas sean redirigidas a la computadora en la red local.
- El servicio de internet se contrate con una IP fija, o bien registrar la IP dinámica automáticamente a través de un servicio como my-ip.com.

1.3 Definición de Objetivos

El Objetivo general y los objetivos específicos se muestran a continuación:

1.3.1 Objetivo General

Diseñar y desarrollar un prototipo de un Sistema Domestico de Video Vigilancia con Cámaras IP, para el monitoreo remoto por internet, para contribuir a la prevención del delito en cuanto al robo de domicilios.

1.3.2 Objetivos Específicos

Los objetivos específicos del presente proyecto son los siguientes:

- Obtener información de proveedores y empresas que ofrezcan el servicio y productos de video vigilancia con cámaras IP en el país.
- Estudiar las diferentes necesidades técnicas para la instalación del sistema en lo referente a las cámaras IP inalámbricas, routers, ancho de banda, resolución de la imagen.
- Diseñar la red para la transmisión de video por internet mediante las cámaras IP inalámbricas.
- Implementar el software de gestión para la manipulación y configuración de las cámaras IP por un navegador web.

- Implementar un prototipo de video vigilancia con dos cámaras IP por internet, que permita el acceso a las cámaras IP por medio de un navegador web.
- Evaluar si las necesidades técnicas planeadas en el sistema de video vigilancia son consistentes con los resultados obtenidos en el prototipo.

2. FUNDAMENTO TEÓRICO

2.1. Introducción

El sistema de video vigilancia con cámaras IP por internet tiene el principio de un networking o una red, es decir, un conjunto de dispositivos que permiten la comunicación y la transmisión de datos por internet, por lo que es necesario conocer los conceptos y tecnologías básicas que integra el networking.

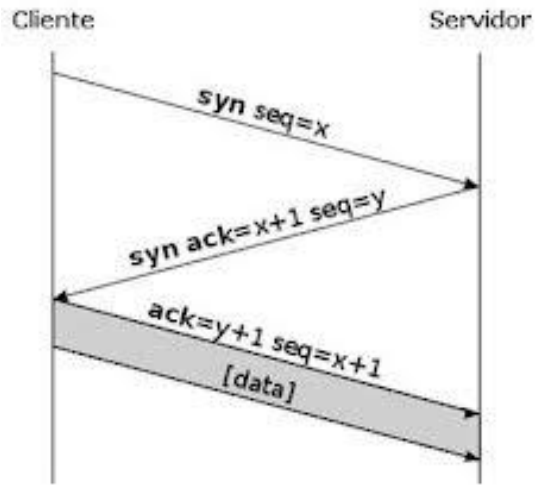
La implementación de redes usan los modelos OSI y TCP/IP que brindan las diferentes funciones y servicios a una red, detallados más adelante, entre estos tópicos encontramos los dispositivos de red, esquemas de direccionamiento y los medios que se usan para la transmisión de la información.

Para el diagnóstico y comportamiento de las redes se utilizan herramientas como el Wireshark, y el Capsa Free Colasoft, que permiten comprender el flujo de los datos en la red y observar comportamientos para obtener patrones de comportamientos. En el caso del presente trabajo, se va analizar el flujo de paquetes que viaja desde la cámara hasta llegar al navegador web, en cuanto a los protocolos utilizados para la transmisión tenemos server push que viene a ser un tipo MIME (multipart-mixed-replace) de HTTP.

HTTP server push, es conocido también como el protocolo HTTP streaming, en la que se considera como un mecanismo para la transmisión de datos desde un servidor web hacia un navegador web. La característica especial de éste protocolo radica en que, "*el servidor web no termina la respuesta después que los datos han sido enviados al cliente"* (Internet, Wikipedia). De esta manera

el servidor web deja el canal abierto en el caso de existir más eventos, pueda seguir enviando datos al cliente. [2]

Así mismo existe otro mecanismo para la implementación de envío de datos mediante este protocolo denominado ***multipart/x-mixed-replace***, introducido por Netscape, en donde los navegadores web interpretan como un cambio en el documento en la que, el servidor pública una nueva versión para el cliente. Es decir existe en la actualidad un soporte para navegadores como Firefox, Opera, Chrome, y Safari, pero a nivel de Internet Explorer no existe soporte para el protocolo server push.[3] Básicamente el viaje de datos con el protocolo server push, se representa en la Ilustración N° 2, en la que como se menciona el envío de datos termina hasta que los datos se envíen al cliente.



Fuente: es.wikipedia.org

Ilustración N° 2. Establecimiento de la negociación en tres vías (SYN, SYN/ACK y ACK).

[2] http://es.wikipedia.org/wiki/Tecnolog%C3%ADa_Push
[3] http://es.wikipedia.org/wiki/Tecnolog%C3%ADa_Push

Para determinar el ancho de banda que utiliza la trasmisión de video de las cámaras se podría usar software más sofisticado como PRTG por sus siglas Paessler Router Traffic Grapher, en su versión free.

2.2 Redes de datos

2.2.1 Generalidades

La existencia humana desde sus inicios buscó la forma de comunicarse con los demás para transmitir sus ideas y mensajes a través de canales adecuados, los mismos que han ido evolucionando con el tiempo hasta consolidar redes de datos que agregan voz, flujo de video, textos y gráficos, esquema que se puede en la Ilustración N°3. Logrando desarrollar además la tecnología Inalámbrica que "*es la que permite la comunicación sin necesidad de contar con conectividad física, como por ejemplo los celulares, asistentes digitales personales o por sus siglas en ingles personal digital assistants* (PDA)"[4]. A partir de la cual podemos tener un acceso remoto a las diferentes aplicaciones como si las personas estuvieran presentes en el lugar. Gracias a estos adelantos tecnológicos es posible la implementación de un sistema de video vigilancia con cámaras IP, que en lo posible serán de tipo inalámbrica.

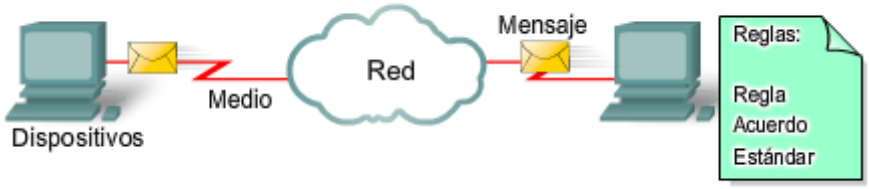
[4] CNNA Exploration, Aspectos Básicos de networking, Glosario.

Fuente: Cisco Networking Academy

Ilustración N° 3. Redes en el mundo.

2.2.2 Elementos de una Red

Dentro de una red, los datos que se envían como mensajes pueden estar contenidos en páginas web, en correos electrónicos, en mensajes instantáneos en de redes sociales, llamadas telefónicas y demás formas de comunicación a través del internet. Estos mensajes necesitan un medio y servicio así como reglas para que permitan la comunicación de los mensajes, con lo que se puede colegir que los elementos básicos de una red son: Dispositivos, Medios, Mensajes, Reglas[5](Ver Ilustración N° 4).



Fuente: Cisco Networking Academy

Ilustración N° 4. Elementos básicos de una red.

A los dispositivos se los puede dividir en: comunes, e intermedios.

[5] CNNA Exploration, Aspectos Básicos de networking, I Elementos de una red.

Los dispositivos comunes que son las PCs de escritorio y portátiles, servidores, teléfonos IP, Cámaras IP, responsables de originar mensajes y que están conectados por medios de una conexión LAN con cables o Inalámbricos o de ser el caso con un enlace WAN.

Los dispositivos intermedios que son los que ayudan a dirigir y administrar los mensajes en la red, entre estos tenemos: Switch para redes LAN, Router que ayuda a dirigir mensajes, firewall que proporciona seguridad a las redes.

En cuanto a los medios por los que viajan los mensajes o datos, tenemos que los dispositivos comunes e intermedios deben estar conectados a la red local conocida como LAN por sus siglas Local Área Network, o expandida conocida como WAN por sus siglas Wide Área Network, por medio de cables que puede ser cobre con señales eléctricas, fibra óptica con señales de luz, o medio inalámbrico con microondas en el espacio.

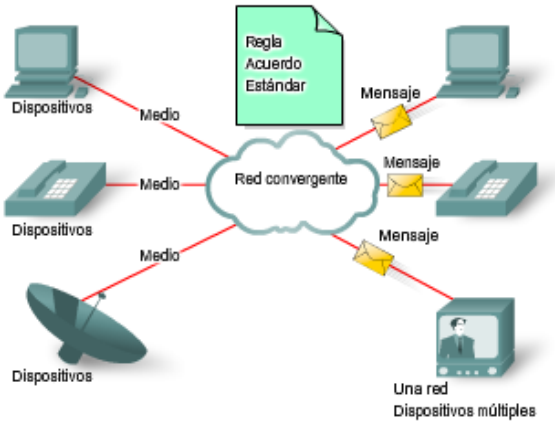
Para el caso del sistema de video vigilancia se van usar dispositivos que envían imágenes por medio de cámaras IP, por medio de enlace LAN, estos datos son enviados a un navegador web previa autentificación de usuarios en el servidor web, para que el cliente de forma remota por medio de un navegador web realice peticiones (URLs o direcciones de las cámaras IP) al servidor web y poder visualizar el video que se trasmite por medio de las cámaras IP.

Las cámaras IP, envían imágenes a través del protocolo conocido como server push, por lo que requiere de servicios y funcionalidad para su manejo y trasferencia, por otro lado al tener una aplicación web con uno o varios clientes se

utiliza el protocolo HTTP para su transferencia de información, que en lo posterior se procede al detalle para su mejor entendimiento.

2.2.3 Redes Convergentes

En el pasado servicios como la telefonía, la radio, la televisión entre otras utilizaban tecnologías diferentes, es decir, un servicio como tal para le telefonía en la que existía sus propias reglas e infraestructura y así con las demás tecnologías. Hoy en la actualidad con el avance de la ciencia se le conoce como red convergente, que significa que la telefonía, la trasmisión de datos se realiza dentro de una misma infraestructura, es decir, flujos de voz, video y datos se transmiten por una misma red, sin la necesidad de separar las tecnologías, es así que a través del servicio de internet se logra obtener datos, televisión. Luego de tener claro la definición de redes convergentes, se puede brindar un servicio adicional de video vigilancia con cámaras IP, incorporando dentro del servicio de Internet con una orientación hacia el monitoreo remoto.[6] (Ver Ilustración N° 5).



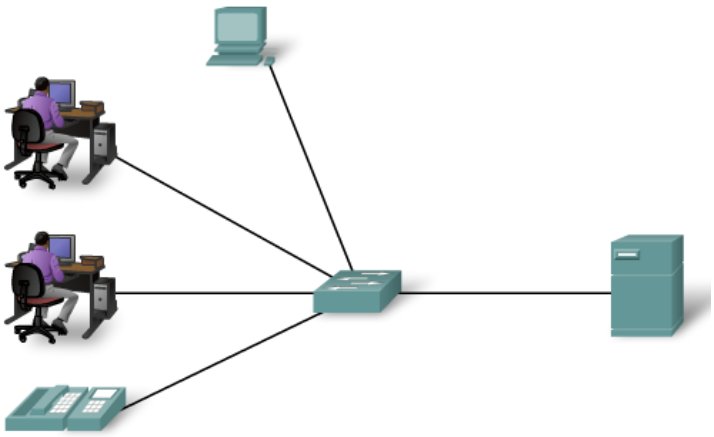
Fuente: Cisco Networking Academy

Ilustración N° 5. Redes Convergentes.

[6] CNNA Exploration, Aspectos Básicos de networking, Capitulo I, Redes Convergentes

La Ilustración N° 5, representa una red convergente en la que varios servicios como el de telefonía, datos, convergen en una sola red llamada internet.

2.2.3 Redes de área local (LAN)



Fuente: Cisco Networking Academy

Ilustración N°6. Red LAN Local Área Network.

El término de “red de área local” o “Local Área Network” LAN, es aquella intercomunicación de dispositivos que se encuentran bajo un mismo nivel de administración. (Ver Ilustración N° 6). En la Actualidad se habla de redes locales interconectadas por cientos de host o estaciones de servicios, las mismas que se instalan en edificio, o dentro de un conjunto de edificios juntos o próximos.

La implementación del prototipo de un sistema de video vigilancia con cámaras IP, se encuentra dentro de una configuración de red local o conocida como LAN que utiliza IPs clase C, la misma que integra las cámaras IP con un dispositivo intermedio como un router o un swtich para su direccionamiento al servidor web.

La configuración del prototipo de sistema de video vigilancia con cámaras IP, inicia con un usuario, el mismo que cuenta con un servicio de internet a través de su proveedor conocidos como (Internet Service Provider) ISP, a partir de la cual cuenta con una red pública de telefonía con un modem o router inalámbrico, una línea digital de abonado denominado (Digital Suscriber Line) que proporciona enlaces de alta velocidad a través de líneas de telefonía con un modem especial DSL. Estos elementos descritos permite la implementación del prototipo, tomando en cuenta una conexión a un corta fuego o del inglés firewall que ofrece servicios de seguridad. (Stallings, 2004). Es importante recalcar que en el caso del prototipo de video vigilancia el firewall viene a constituirse el router inalámbrico.

Previo a la elección de la configuración de la red LAN o local, para el sistema de video vigilancia se analiza las características entre una red LAN inalámbrica, red LAN Ethernet de alta velocidad. Los principales elementos de una red son; topología, medio de transmisión, disposición técnicas de control de acceso al medio, que permite determinar el costo, la capacidad, el tipos de datos a transmitir, la velocidad y la eficiencia de las comunicaciones. (Stallings, 2004).

La topología a utilizar en el sistema de video vigilancia es la topología de estrella. "*La topología de estrella cada estación está directamente conectada a un node central común, generalmente a través de dos enlaces punto a punto, uno para transmisión y otro para recepción*". (Stallings, 2004). Esta topología aprovecha de forma natural el cableado de las instalaciones en edificios, mejor para distancias cortas, además de ofrecer velocidades elevadas a un número pequeño de dispositivos. (Stallings, 2004). Para el caso del prototipo en estudio

se considera de tipo estrella porque las cámaras IP están conectadas al router en la que se conectan a una red LAN.

En la elección del medio de trasmisión se recomienda el uso de cable (Unshielded Twisted Pair) UTP o conocido como Par Trenzado no Apantallado tipo 5 que permite trasmitir velocidades de hasta 100 Mbps a frecuencias de hasta 100MHz y da mayor rendimiento que se acopla bien con la topología de estrella. El sistema de video vigilancia en lo posible utiliza cámaras IP inalámbricas. (Stallings, 2004).

Los enfoques que se utilizan para el diseño de redes LAN de alta velocidad se muestran a continuación en la siguiente Tabla N° 1.

	Fast Ethernet	**Gigabit Ethernet**	**Canal de fibra**	**LAN inalámbrica**
Velocidad de datos	100Mbps	1 Gbps,10Gbps	100Mbps-3, 2Gbps	1 Mbps-54 Mbps
Medio De transmisión	UTS, STP, Fibra óptica	UTP, cable apantallado Fibra óptica	Fibra óptica Cable coaxial, STP	Microondas 2,4 GHz, 5 GHz
Método de acceso	CSMA/CD	Conmutado	Conmutado	CSMA/Sondeo
Estándar	IEEE 802.3	IEEE 802.3	Asociación del canal de fibra	IEEE 802.11(b/g/n)

Fuente: Stalling Redes LAN de alta velocidad

Tabla N° 1. Características de algunas redes LAN de alta velocidad.

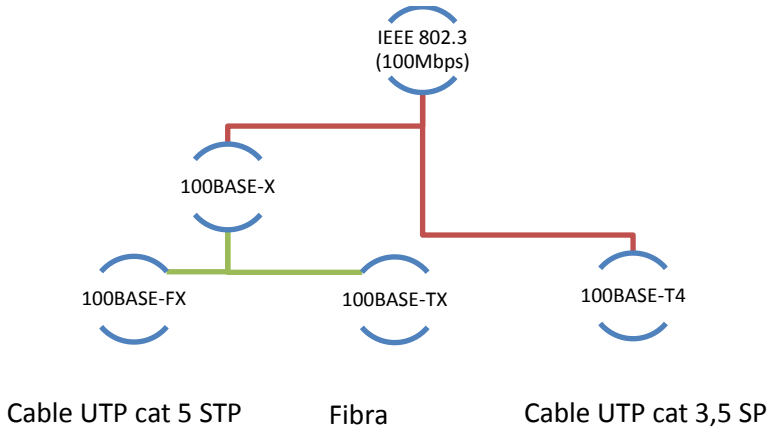
La tecnología Ethernet está basada en el estándar desarrollado en la IEEE 802.3, que establece las funcionalidades y compatibilidades con los dispositivos. Aquí se menciona la técnica CSMA, Carrier Sense Multiple Access que generalmente se basa en determinar si el medio de transmisión está siendo

usada, si es así existe una espera, y si existe una transmisión mutua entre estaciones se produce colisiones. Se deja claro que existen técnicas CSMA para contrarrestar los problemas de transmisión colisiones.

Básicamente es necesario conocer como el video de las cámaras IP tienen que viajar por el medio, para lo cual se toma en cuenta las observaciones hechas por esta técnica para evitar las colisiones en la transmisión haciendo que el video no llegue a tener paros en su secuencia de imágenes. (Stallings, 2004).

La normalización que ofrece IEEE 802.3 está definida por:

10BASE5, 10BASE2, 10BASE-T, 10BASE-FP, para Ethernet, en donde existen alternativas con diferentes versiones que funcionan a 100Mbps, 1 Gbps, y 10 Gbps. Para Fast Ethernet que es una tecnología que funciona a 100Mbps también se restablece alternativas para estándares 100BASE-T, de la cual se resumen en la siguiente gráfica (Ver Ilustración N° 7). (Stallings, 2004).



Fuente: Generación propia.

Ilustración N° 7. Opciones 100BASE-T en IEEE 802.3.

Como se muestra en la Tabla N° 1, se encuentran otras alternativas las mismas que de una forma general las mencionamos a continuación:

GIGABIT ETHERNET es una de las alternativas para 1 Gbps, 1000BASE.SX, 1000BASE-LX, 1000BASE-CX, 1000BASE-T.

ETHERNET de 10 Gbps es lo nuevo, es decir, esta tecnología se ha desarrollado por el incremento de tráfico en el internet e intranets, haciendo que los ISP, Internet Service Providers den un enlace de alta velocidad a un costo reducido, que permite además más distancia que van de desde los 10 metros hasta los 100 kilómetros. (Stallings, 2004).

En la actualidad el servicio de internet está bajo 100BASE-T y en algunos casos en 100BASE-FX, en la que la velocidad de trasmisión es de 100Mbps con una longitud de segmento de 100 metros y cobertura de la red entre 1000 y 400 metros, en el caso del prototipo de video vigilancia usaremos lo que establece IEEE 802.11, para el caso de usar cámaras IP inalámbricas en el exterior y el estándar IEEE 802.3 para cámaras internas con cableado, para mejorar la confiabilidad del servicio. (Stallings, 2004).

Por último para el prototipo del sistema de video vigilancia se utiliza redes LAN inalámbricas que se encuentran en un avance en el mercado de redes locales, ya que permite movilidad, traslados, trabajo en red ad hoc además de cobertura en lugares de dificultad para el cableado. Existen edificios, locales comerciales, empresas que poseen cableado de par tranzado y en algunos casos el mantenimiento resulta oneroso, es por esto que la LAN inalámbrica resulta una alternativa efectiva y más atractiva, sin olvidar que siempre va a ver conexiones

LAN cableadas e inalámbricas en conjunto dependiendo de las funciones que se necesiten. La especificación para las redes LAN inalámbricas se encuentra bajo la IEEE 802.11 b/g/n que va a depender del soporte que poseen las cámaras IP.

Una red LAN inalámbrica cumple con los requisitos para cualquier otra LAN, así por ejemplo debe cumplir con capacidad de difusión, cobertura a distancias pequeñas, conectividad entre otras. Para el caso del sistema de video vigilancia se toman las siguientes consideraciones:

- Buscar un rendimiento del medio inalámbrico para maximizar la capacidad.
- La área de servicio necesaria para una red inalámbrica está entre 18 y 38 metros aproximadamente, de la cual consideramos que las cámaras van estar localizadas dentro del rango.
- El Consumo de Energía de las estaciones de trabajo de las cámaras utiliza adaptadores.
- Robustez en la transmisión y seguridad en la que de acuerdo al diseño de la red LAN, debe permitir trasmisiones fiables en entornos de ruido y controlar las posibles interferencias para lograr un nivel de seguridad.
- Controlar redes adyacentes que se refiere a redes que operan en la misma zona y que exista una interferencia para lo cual es necesario asegurar que se encuentre en un mismo canal para evitar interferencias previo configuración automática que se realiza. (Stallings, 2004).
- A continuación el modelo IEEE 802.11 presenta los tres servicios que proporciona una red LAN inalámbrica:

- Autenticación.- Para el caso de una red LAN cableada el acceso a una conexión física tiene una conectividad, para el caso de la red LAN inalámbrica se requiere de una antena debidamente sincronizada.
- Fin de la autenticación.- Cuando el servicio es invocado siempre que se vaya a dar por finalizada una autenticación existente.
- Privacidad.- Este punto es importante para el sistema de video vigilancia ya que este servicio asegura que los contenidos de los mensajes no sean leídos por personas o dispositivos ajenos, en la que el estándar incluye mecanismos de privacidad como el cifrado.

2.2.4 Modelo OSI y TCP/IP

MODELO OSI
Aplicación Capa 7 Proporciona el acceso al entorno OSI y da servicios a las aplicaciones, servidor de correo, presentar datos, codificación y conversión de formatos.
Presentación Capa 6 Proporciona formato y compresión de los datos, sintaxis.
Sesión Capa 5 Proporciona el control de la comunicación, establece, gestiona y cierra sesiones.
Transporte Capa 4 Proporciona transferencia confiable de datos entre los puntos finales, y procedimientos de recuperación de errores y control. Segmentation. (PDU segmento)
Red Capa 3 Proporciona independencia a los niveles superiores respecto a técnicas de transmisión para conectar sistemas, direccionamiento lógico con las IPs, Enrutamiento buscando el mejor camino. (PDU paquete)
Enlace Capa 2 Direccionamiento físico, conoce la MAC address. (PDU trama)
Física Capa 1 Transmisión de cadenas de bits sobre el medio físico, características mecánicas, eléctricas. (PDU bits)

Fuente: Stalling Redes LAN de alta velocidad

Tabla N° 2. Las capas OSI.

Según la Tabla N°2, es necesario hacer mención de las arquitecturas de protocolos, que determinan los estándares de comunicación, tenemos el conjunto de protocolos TCP/IP que es la más usada y el modelo de referencia OSI que se utiliza como modelo educativo para la comprensión de las redes de comunicaciones. (Stallings, 2004).

En resumen se puede decir que el sistema con sus siete capas interactúan entre sí, inicia cuando una web por ejemplo a través de sus aplicación desea transmitir un mensaje este invoca a la capa de aplicación (Capa 7), luego este protocolo usa los servicios de la Capa 6, usando protocolos en común, y así sucesivamente hasta llegar a la capa física o Capa 1, en la que es realmente donde se transmiten los bits a través del medio físico, como li indica la Tabla N°2.

2.2.5 Modelo TCP/IP

La arquitectura de protocolos TCP/IP es un desarrollo tecnológico que ha permitido establecer los estándares de internet que es utilizado en el campo real de las comunicaciones y está definida por parte de IAB, Internet Architecture Board. El modelo contempla cuatro etapas a diferencia del modelo OSI.

- Capa de acceso a la red
- Capa internet
- Capa de transporte
- Capa de aplicación

Al igual que el modelo OSI, la capa física es la que se encarga de las especificaciones y características del medio de transmisión.

Dentro de este modelo la capa de acceso a la red es responsable del intercambio de datos entre un servidor o estación de trabajo y la red a la que está conectado. El usuario proporciona a la red la dirección de destino, para que se pueda encaminar los datos hacia el destino final.

La capa de internet IP Internet Protocol es aquel que permite ofrecer servicios de enrutamiento a través de varias redes, este enrutamiento conocido también como encaminador es el que ayuda a retrasmitir datos desde una red a otra por una ruta adecuada.

La capa de transporte es utilizada para el control de la transmisión, TCP, Trasmission Control Protocol.

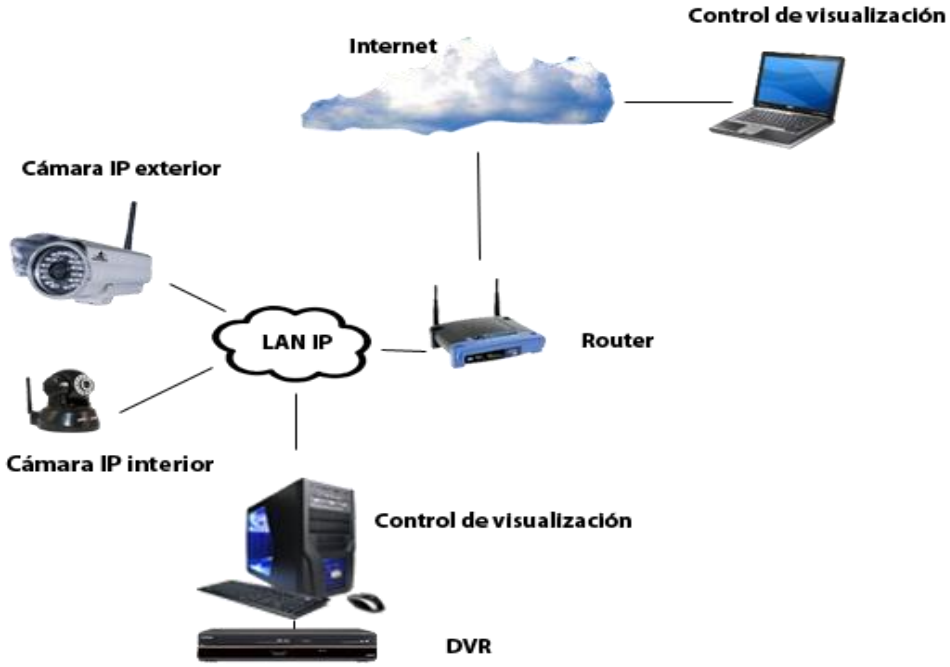
Finalmente la capa de aplicación es la que permite al usuario una transferencia de archivos por ejemplo. Todo lo antes mencionado, se encuentra presentado de una forma general y especifica, que permite tener un mejor conocimiento de la tecnología de las comunicaciones, y poder analizar los paquetes que las cámaras IP envían a través de las redes, verificar el protocolo que utiliza para la transferencia.

2.3. Cámaras IP

“Cámara de red o IP es un dispositivo que capta y transmite una señal de audio/video digital a través de una red IP estándar a otros dispositivos de red, tales como un PC, un teléfono inteligente. Mediante el uso de una dirección IP, un servidor web y protocolo de streaming de video, los usuarios pueden visualizar, almacenar y gestionar video de forma local o remota“(García, 2010).

Las cámaras IP van a permitir al usuario la visualización en tiempo real de los dispositivos de red, y además se puede monitorear, modificar movimientos de cámaras precautelando los bienes, siendo por lo tanto un sistema más cómodo y con más prestaciones que los sistemas de vigilancia con cámaras análogas que se utilizan en circuitos cerrados de televisión.

La arquitectura básica de un sistema de video vigilancia por internet con cámaras IP es el siguiente: servidor web, servidor de almacenamiento, elementos de la red de datos como cableados, routers, switches, según la Ilustración N° 8 que se muestra a continuación.

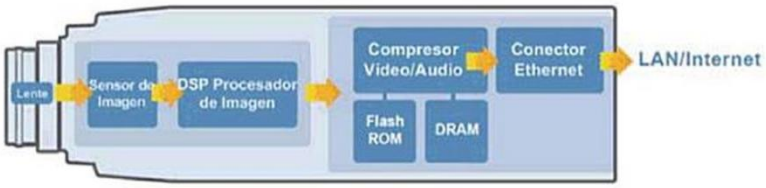


Fuente: Generación propia

Ilustración N° 8. Esquema de un sistema domestico de video vigilancia IP.

2.3.1 Componentes de Cámaras IP

Internos.- De una forma muy general se presenta los componentes que integran una cámara IP, la misma que puede variar de acuerdo a su precio y funcionalidad adicional que pueda brindar al cliente o usuario. Posee un lente, un sensor de imágenes, un procesador de imágenes, un chip de compresión de video, y un chip de Ethernet que permite conectividad de red para la transmisión de datos, esto se puede apreciar en la Ilustración N° 9, que se indica a continuación. (García, 2010).



Fuente: Francisco Javier García Mata

Ilustración N° 9. Esquema básico de una cámara IP interna.

Externos.- Una cámara IP básicamente tiene los siguientes componentes que permiten su configuración dentro de un sistema de video vigilancia. (Ver la Ilustración N° 10)

Fuente: Generación propia

Ilustración N° 10. Componentes externos de una cámara IP.

Una cámara IP pude además ser de diferente tipo de acuerdo su uso, así por ejemplo para interiores y exteriores, se clasifican en cámaras de red fijas, domo fijas, PTZ (Pan-Tilt-Zoom) y domo PTZ.

Las cámaras de red fijas como su nombre lo indica son las que permanecen fijas hacia un objetivo, ideales para exteriores en puntos considerados como importantes o críticos de un lugar determinado.

Las cámaras de red domo fijas, las mismas que son cámaras fijas pero dentro de una carcasa, a prueba de vandalismo que se instalan de forma predeterminada en techos, su característica principal es su discreción y resistente a las manipulaciones que puedan ser objeto. Su limitado espacio no permite en ciertos casos cubrir varios objetivos, por lo que resulta útil para objetivos fijos.

Las cámaras de red PTZ, Pan.Tilt.Zoom, que permiten moverse de forma horizontal y verticalmente, además de poseer un zoom ajustable dentro de una determinada área, su utilización es ideal para espacios amplios para cubrir varios objetivos. Este tipo de cámaras puede incorporar las siguientes funcionalidades:

- Estabilización electrónica de imagen (EIS).- Ayuda a reducir el efecto de vibración de un video, además de reducir el tamaño de la imagen comprimida, haciendo de suma utilidad al momento del almacenamiento.
- Posiciones predefinidas.- Permite configurar posiciones predefinidas, haciendo que el operador pueda cambiar de una posición a otra de forma rápida.
- E-flip.- Permite para realizar seguimientos de personas cuando esta pase por debajo de la cámara.

- Auto-flip.- Permite girar la cámara 180 grados de su cabezal y realizar un movimiento horizontal enfocándose en un algunos casos en un objetivo.
- Autoseguimiento.- Permite la detección de movimiento de un objeto y lo sigue dentro de la cobertura de la cámara, funcionalidad que permite la efectividad en áreas de grabación de una escena en actividad.

2.3.2. Características de Cámaras IP

- Procesador 32 Bit RSCIC incorporado
- Protocolos TCP/ID, UDP, IMCP, SMTP, HTTP, FTP, DHCP, PPPoE
- Formato de compresión con M-JPEG.(Envío de imágenes comprimidas)
- Wifi incorporado
- Control de Movimiento en un rango de 270° horizontalmente y 120” verticalmente.
- Audio de 2 vías.
- Sensor CMOS ¼ pulgadas de color.
- Distancia de 5-10 metros de visión nocturna.
- Configuración en redes LAN/WAN/Internet.
- Captura y Grabación de video en tiempo real y remoto desde PC.
- Detección de movimiento.
- Alertas vía email.
- Encriptación WIFI WEB, WPA y WPA2
- Estándar WIFI 802.11b/g

2.3.3. Ventajas de la Video vigilancia IP

Entre las más importantes podemos mencionar la accesibilidad remota, la calidad de imagen, la gestión de video mediante eventos, capacidades de almacenamiento, integración de tecnologías, escalabilidad, flexibilidad, y rentabilidad.

Con el acceso remoto los usuarios previamente autorizados pueden visualizar el video en tiempo real, en cualquier ubicación y en cualquier momento. A diferencia de un sistema de vigilancia de circuito cerrado por sus siglas CCTV. En cuanto a la imagen las resoluciones de las cámaras IP están sobre las de una cámara analógica utilizada en CCTV o circuito cerrado de video vigilancia, ya que las imágenes son digitalizadas sin una conversión y se pueden almacenar y recuperar en una computadora. El manejo o gestión del video a través de un software de gestión, que incluye funciones como detención de movimiento, alarmas de detención, conexiones de entrada y salida E/S.

Un sistema de video vigilancia permite una escalabilidad y flexibilidad, al poder incorporar un determinado número de cámaras IP, ya sea de forma inalámbrica o con cable.

2.3.4 Aplicabilidad de las cámaras IP

Antes de implementar un sistema de video de vigilancia con cámaras IP se debe tomar en cuenta que el prototipo puede tener funciones de vigilancia en en tiempo real, monitoreo de video, control de alarmas y análisis de imagen, y en algunos casos almacenamiento de video.

El sistema de video vigilancia puede tener un gran número de cámaras IP, para lo cual la arquitectura a utilizar puede ser cliente/servidor. Las soluciones clientes/servidor basadas en arquitecturas TCP/IP ya sea bajo un entorno LAN o WAN, permiten que sea implementado en diferentes plataformas y que exista una estabilidad del sistema. Los usuarios pueden utilizar sistemas operativos que soporten Java, además que puedan ser capaces de acceder a las imágenes desde cualquier ubicación utilizando un navegador web, agregando funciones de seguridad como encriptación de los datos, red privada virtual.

Requerimientos para ejecutar la aplicación web dentro del sistema de video vigilancia.

- Servidor de TOMACAT mínimo versión 6.0
- Base de datos MySQL en la que por medio de Hibernate permite Persistencia.
- Dominio (ecuavisor.com)
- Hosting
- Sistema Operativo de preferencia LINUX o WINDOWS.

2.4. Suite TCP/IP y Server push

2.4.1 Protocolo HTTP

"El Protocolo de Transferencia de HiperTexo (Hyper Transfer Protocol) es una sencillo protocolo cliente-servidor, que articula los intercambios de información entre los clientes Web y los servidores HTTP. La especificación completa del protocolo HTTP 1/0 está recogida en el RFC 1945." (ROMERO, 203)

Es un protocolo sencillo que da soporte para servicios de conexión TCP/IP, además de funcionar en soporte para entornos UNIX: "*un proceso servidor escucha en un puerto de comunicación TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores*." (ROMERO, 203)

Es decir el protocolo se basa en operaciones de solicitud/petición, en la que un cliente establece una conexión con un servidor y envía mensajes con datos en la solicitud, a partir de la cual el servidor responde con un mensaje en la que específica el estado de la operación y el resultado de la petición que realiza el cliente.[7]

Entre las características principales se menciona las siguientes:

- La comunicación se realiza en modo binario con caracteres de 8 bits.
- Transferencia de objetos multimedia, identificado por su clasificación MIME.
- Los procesos que utiliza el protocolo HTTP para el dialogo con el servidor son: GET que recoge el objeto, POST para enviar la información al servidor y HEAD para solicitar características de un objeto, por ejemplo la fecha.(ROMERO, 203)

2.4.2 Server push

Server push es un concepto basado en MIME, que permite que un servidor envíe datos hacia el cliente manteniendo una conexión abierta. HTTP

[7] Publicar en Internet: Guía Práctica para la creación de Documentos HTML.

normalmente permite peticiones desde un cliente hacia el servidor, pero no el caso contrario. Al mantener una conexión abierta, el servidor puede enviar notificaciones al cliente.

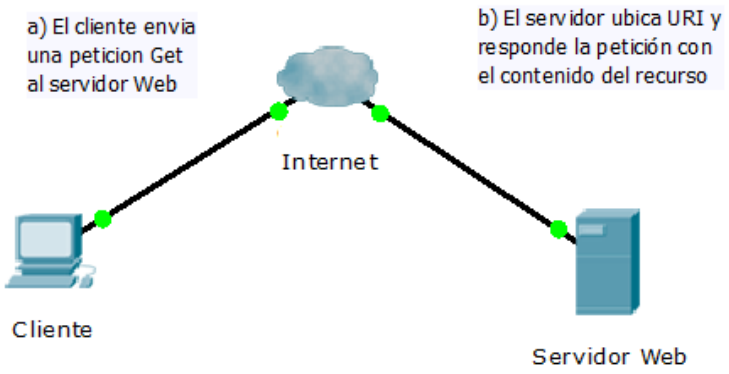
Para el caso de las cámaras, el video es en realidad una secuencia de imágenes que se envían mediante el protocolo Server push, usando el tipo MIME multipart-mixed-replace. Mediante Wireshark podemos ver la petición inicial de las imágenes y cómo la cámara envía una a una las imágenes como un stream.

2.5. Aplicaciones Web

2.5.1 Transacciones HTTP

Las aplicaciones Web contienen un lenguaje de marcado de hipertexto extensible (XHTML), es decir contiene etiquetas para que el navegador pueda mostrar y presentar la información, además de incluir comandos para el servidor, imágenes y datos binarios. En aplicaciones web contienen secuencias de código de Cascading Style Sheets (CSS), AJAX. . (Deitel, 2008)

HTTP contiene los métodos y encabezados para que el cliente pueda interactuar con el servidor, intercambiando información, utiliza los denominados URIs (Identificadores uniformes de recursos) los cuales especifican las ubicaciones de los datos u objetos que interactúan o procesan datos a través de la base de datos.



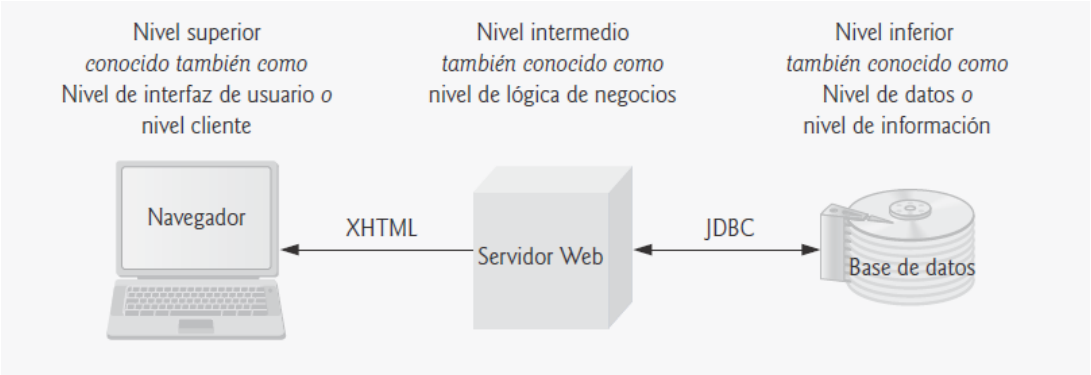
Fuente: Generación propia

Ilustración N° 11. Interacción entre el cliente y el servidor Web.

En la Ilustración N° 11, se indica una representación gráfica de una transacción HTT, en el que se observa la interacción del cliente el internet y el servidor, ante una solicitud y respuesta de una operación que se necesite.

2.4.2 Arquitectura de tres niveles.

La arquitectura de tres niveles se refiere, cuando las aplicaciones Web tienen diversas funcionalidades, en niveles separados permitiendo tener el control de cambios en sus funcionalidades, como puede ser acceso a la interfaz de usuarios, a la base de datos o algún servidor según indica en la Ilustración N° 12.



Fuente: How to Programming Java, Deitel.

Ilustración N° 12. Arquitectura de tres niveles en aplicaciones Web.

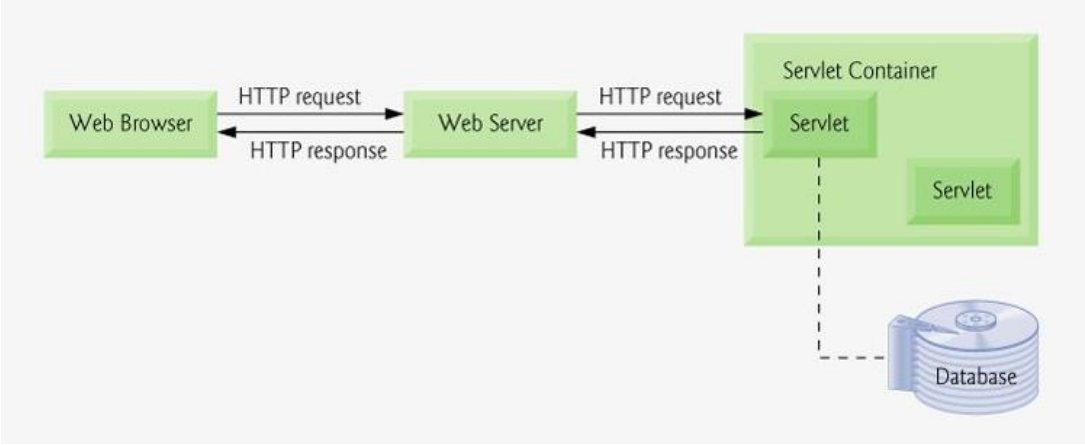
Un nivel inferior que contiene los datos e información en una base de datos por ejemplo, un nivel intermedio o de lógica del negocio que procesa las peticiones de los clientes y obtiene datos de la base de datos para presentarle al cliente en forma de documento XHTML. Un nivel superior que es la interfaz de la aplicación que le permite al cliente recopilar datos de entrada y de salida.

La aplicación web que se implementa para el prototipo de video vigilancia está desarrollada mediante el lenguaje de programación java con Java EE, Java Enterprise Edition, permitiendo estas funcionalidades descritas a través de sus niveles. (Deitel, 2008)

2.4.3 Tecnologías Web de Java

2.4.3.1 Servlets

Tecnología Java que permite extender funcionalidad de un servidor, generando contenido dinámico, que se utiliza en un entorno de aplicación web. Los servlets permite acceso seguro a un sitio Web, interactuar con la base de datos, proporciona información de sesiones de clientes. (Ver Ilustración N° 13).



Fuente: How to Programming Java, Deitel.

Ilustración N° 13. Arquitectura Servlet.

El contenedor de `servlets` es el encargo de ejecutar los mismos, recibe las peticiones HTTP de un cliente y dirige la petición al servlet, este procesa la petición y devuelve una respuesta en forma de documento XHTML o XML (Lenguaje de marcado extensible), a través del navegador web.

XML se le conoce como un lenguaje que permite el intercambio de datos estructurados en la Web. [8]. En donde para el prototipo implementado se utiliza su equivalente que es JSON que es el formato para el intercambio de datos.

Los servlet implementa la `interfaz Servlet` del paquete `javax.servlet`. Esta interfaz declara métodos para el contenedor de servlet y administrar el ciclo de vida del servlet. El contenedor `servlet` invoca el *método* `init` del servlet una sola vez durante el ciclo, quedando listo para una respuesta. El método `service` de un servlet maneja las peticiones, que recibe la petición, la procesa y envía la respuesta, existiendo una llamada al *método* `service` por cada petición. EL *método* `destroy` del servlet libera recursos para obtener mejor memoria. (Deitel, 2008)

La *interfaz* `Servlet` contiene métodos que se invocan automáticamente por el servidor, y son los siguientes:

[8] Capítulo 26.4.1 Servlets Pág. 6

Método	Descripción
void init(ServletConfig config)	Inicializa el servlet, el argumento suministra el contenedor servlet.
ServletConfig getServletConfig()	Devuelve una referencia a un objeto que proporciona acceso a la información de parámetros de inicialización del servlet.
String getServletInfo()	Para obtener información del autor y la versión del servlet.
void service(ServletRequest request, ServletResponse response)	El contendor se servlet invoca este método para responder una solicitud de cliente al servlet.
void destroy()	Método que se invoca cuando termina el servlet, liberando recursos utilizados por el servlet.

Fuente: How to Programming Java, Deitel.

Tabla N° 3. Métodos de la interfaz Servlet.

Los Servlet extiende de la `Class HttpServlet` es la que define los métodos `doGet` y `doPost` para las respuestas al cliente, estos métodos se invocan por el método `service`. Existen dos tipos comunes de `HTTP request` como `get` y `post`. Un `get request` es el que obtiene información del servidor y un `post request` envía datos al servidor, como información de autentificación ente otros. . (Deitel, 2008)

Los métodos `doGet` y `doPost` aceptan como argumento un objeto que implementan la interface `httpServletRepuest` y `HttpServeltResponse`, que básicamente permiten la comunicación entre el cliente y el servidor, haciendo fácil el acceso de los datos suministrados por la solicitud y el resultado del servlet al cliente Web, .(Deitel, 2008)

2.4.3.2 Java Server Pages

Es una extensión de la tecnología de los servlet, en donde un contenedor JSPs traduce cada JSP en un servlet, la característica principal es la de separar la presentación del contenido, creando contenido dinámico mediante la reutilización de componentes definidos y secuencias de comandos en el lado del servidor. Al utilizar JSPs se tiene componente como JavaBeans que es reutilizable para el diseño de clases siguiendo un conjunto de convenciones o reglas. Permite el uso de bibliotecas personalizadas que encapsulan cierta funcionalidad dinámica y compleja, como por ejemplo ocultar código para un acceso a la base de datos. (Deitel, 2008)

2.4.4 Persistencia

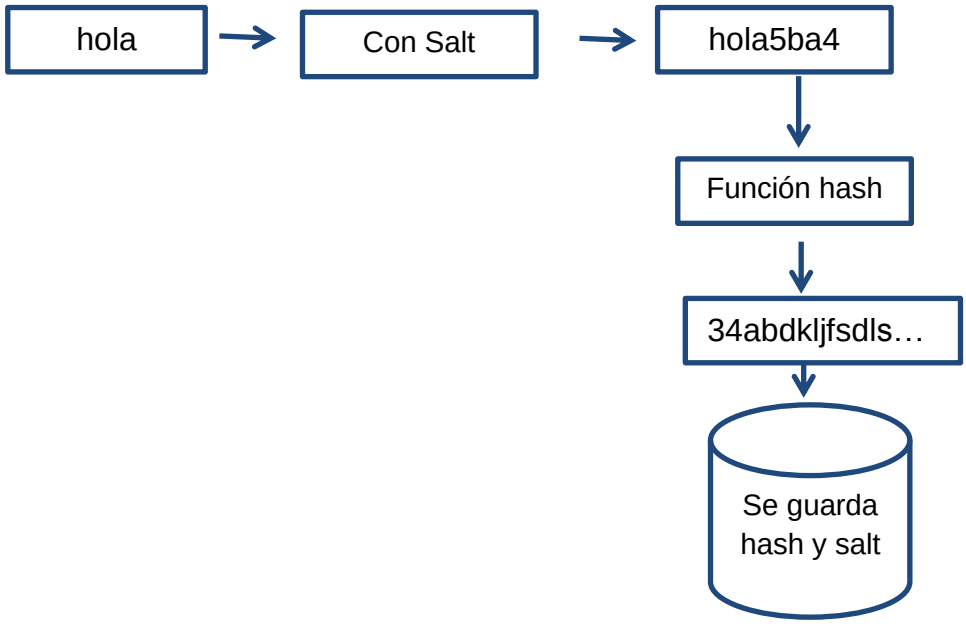
"La persistencia es la capacidad del programador para conseguir que sus datos sobrevivan a la ejecución del proceso que los creo, de forma que puedan ser reutilizados en otro proceso. Cada objeto, independientemente de su tipo, deberá poder llegar a ser persistente sin traducción explicita. También, debería ser implícito que el usuario no tuviera que mover o copiar los datos expresamente para ser persistentes".[9]

En el prototipo utilizamos la persistencia en la manera de convertir los datos relacionales en objetos.

[9] Persistencia de Objetos. JDO, Solución Java, Juan Mármol Castillo tomado de la URL dis.um.es

2.5 Utilitarios y tecnología de soporte

- El SHA (Secure Hash Algorithm o Algoritmo de Hash Seguro).-



Fuente: Generación propia

Ilustración N° 14. Utilización de Salt en una contraseña.

Algoritmo tipo hash utilizado para la encriptación de la información, este tiene variantes como SHA-1, SHA-2. SHA-1 permite hasta 2e64 bits y devuelva una cadena de 160 bits. El utilizado para el prototipo es el denominado SHA-2 o 256, que tiene una longitud de salida de 256, dando como resultado una contraseña de 64 caracteres en base 64, y 512 bits.[10] La característica radica en que no es posible obtener el mensaje original a partir de su hash. Este permite en el sistema de video vigilancia verificar la integridad de los datos, siendo menos vulnerable a los ataques. (Ver Ilustración N° 14)

[10] Seguridad Informática ED.11 Paraninfo pág. 105

- La Salt es una secuencia de bytes, añadida a la contraseña antes de la misma, que se guarda de forma oculta, esto hace que la contraseña sea diferente y proteger contra ataques, ya que elimina la posibilidad de encontrar hash pre calculados.
- Eclipse Link es un conjunto de solución de código abierto para el manejo de persistencia en Java, XML, base de datos y servicios web. Es decir proporciona solución basada en Object-Relational Mapping (ORM Relación-Objetos) JPA, con funciones avanzadas para el mapeo y acceso a la base de datos relacionales.[11]
- JDBC es un API que se encuentran incluidos para J2SE y J2EE, que provee conectividad a la base de datos y acceso a data sources. Para la cual utilizamos únicamente el driver ya que la gestión de los datos se realiza por medio de la persistencia.
- La base de datos que se utiliza es MySQL, que es la más popular en cuanto a código abierto se conoce, disponible bajo el modelo "dual licensing", es decir bajo licencias GNU General Public License conocida como "GPL" y bajo licencia comercial.[12] MySQL proporciona conectividad en aplicaciones desarrolladas en Java a través de un driver MySQL Connector/J, el mismo que para el presente proyecto o prototipo es mysql-connector-java-5.1.22.tar.gz.
- La librería Apache Commons Códec proporciona implementaciones comunes de codificadores y decodificadores tales como Base64, Hex,

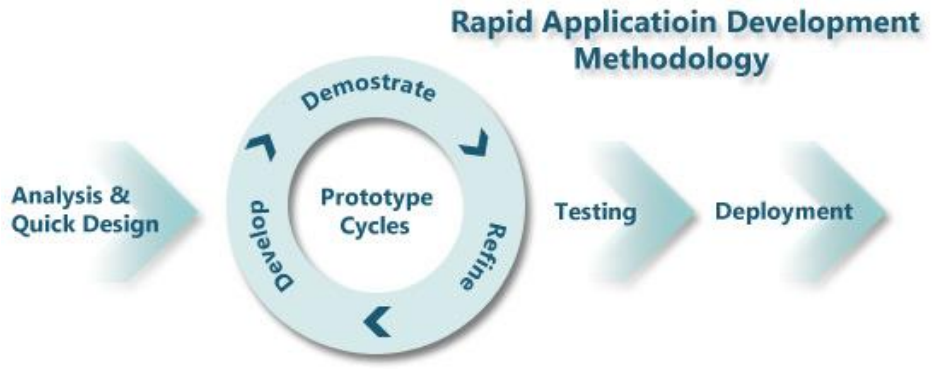
[11] http://www.eclipse.org/eclipselink/
[12] http://www.developer.com

Phonetic and URLs.[13] La versión utilizada es commons-codec-1.7. Esto nos sirve para realizar el Hash de las contraseñas.

- Lombok es una librería que permite generar código en base a las anotaciones, que se integra con javac, Ant, Maven y con Eclipse.[14]

2.6 Metodología

Para el desarrollo del sistema se utiliza la metodología iterativa RAD, Rapid Application Development (Desarrollo Rápido de Aplicaciones), la misma que implica un desarrollo rápido con un fin de alta calidad y la construcción de prototipos. Esta metodología permite un aumento de la productividad, reducción de los costos de desarrollo y con un entregable de calidad. (Ver Ilustración N° 15).



Fuente: How to Programming Java, Deitel.

Ilustración N° 15. Metodología RAD.

El sistema doméstico del presente proyecto encaja dentro de esta metodología al considerar el sistema de video vigilancia doméstico como un proyecto de evolución es decir que puede cambiar en la medida de las

[13] http://commons.apache.org/codec/
[14] http://projectlombok.org

exigencias, y de un entregable de calidad a bajo costo de inversión. La metodología puede incluir herramientas de Interfaz Gráfica de usuario GUI a través de Servlet, para el caso del prototipo de video vigilancia en su inicio se realiza simplemente con lenguaje HTML a través de plantillas ofrecidas en la web, sistemas de gestión de base a través de MySql, lenguaje de programación de cuarta generación que para el proyecto se utiliza JAVA.

EL modelo RAD consiste delos siguientes pasos:

Planificación de requisitos.- Basado en la recolección de requisitos con técnicas de lluvias de ideas que permitan conocer las funciones del sistema, y la forma que soluciona determinado problema.[15]

Descripción del usuario.- Los requerimientos son detallados mediante un feedback con los usuarios construyendo un prototipo usando herramientas de desarrollo. [16]

Construcción.- El prototipo se redefine para construir el producto final y entregar un release al cliente. [17]

Cut over.- En la que incluye la aceptación de las pruebas por el usuario y el entrenamiento. [18]

En resumen se puede resumir la metodología de desarrollo del sistema de video vigilancia mediante el siguiente gráfico.

[15] Software Engineering, Sabharwal Sangeeta, pág 19.
[16] Software Engineering, Sabharwal Sangeeta, pág 19.
[17] Software Engineering, Sabharwal Sangeeta, pág 19.
[18] Software Engineering, Sabharwal Sangeeta, pág 19.

3. DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA

3.1. Análisis y Planificación de requisitos

- El sistema de video vigilancia con cámaras IP por internet debe poder ser accesible desde cualquier parte fuera de la red local implementada y de una forma segura para el usuario a fin de evitar interferencias en la red.
- El sistema requiere integrar de manera adecuada las cámaras IP con la topología de red utilizada y además el flujo de datos deberá pasar por una aplicación web que hace las funciones de servidor web que responde las solicitudes hechas por el usuario, estas solicitudes corresponde a la autentificación de usuario para luego direccionar a las cámaras .
- Además el sistema debe tener la capacidad de crecer en el número de cámaras y tener módulos extras para otros propósitos como por ejemplo manejo de alarmas, envío de emails para alertas.
- El sistema de video vigilancia con cámaras IP, está activo en momentos que el lugar destinado para el sistema esté deshabitado con la finalidad de cumplir con el objetivo de monitoreo remoto desde fuera del domicilio, pero de ser el caso el usuario puede mantener las 24 horas activo el sistema.
- El sistema de video vigilancia con cámaras IP, debe sacar provecho del ancho ofrecido por el ISP o proveedor de internet, reduciendo de esta manera los costos que demandaría un sistema de seguridad privado.
- El registro se realiza mediante la creación de usuarios en la que, un administrador del sistema previo registro del usuario va permitir ingresar los datos del cliente y de esta manera pueda acceder a las cámaras.

- El usuario tiene la posibilidad de mejorar la percepción de seguridad de su domicilio y optimizar los recursos tecnológicos que posee como son: el modem, router, y el servicio de internet.
- El usuario debe poseer un servicio de internet con un router inalámbrico para poder incorporar a la red cámaras IP, así mismo el usuario debe poseer un identificador de usuario, es decir un login a la aplicación web que le permita acceder al contenido de las direcciones IP de las cámaras y poder conectarse remotamente a las cámaras IP instaladas en el lugar asignado para el efecto.
- Las cámaras implementan el protocolo server push, por lo que para poder escalar el prototipo es necesario implementar la clase server push existente para tecnología Java, pero al momento de la trasmisión de las imágenes puede provocar pantallazos más seguidos
- El envío de las imágenes que las cámaras emiten son de tipo streaming con formato HTTP Context Type de tipo multipart-mixed-replaced, siendo de esta manera el navegador quien interpreta esta secuencia de imágenes denominado stream.
- A través del router se puede configurar los puertos por los cuales las cámaras podrán trasmitir la secuencia de imágenes.
- Se utiliza un direccionamiento IP estático mediante la configuración del servicio DHCP, lo cual permite que las cámaras siempre tengan la misma dirección IP que es configurada en un inicio.
- Es decir, el sistema por último propende a la optimización y utilización de los equipos y hardware existentes, previo a las especificaciones dadas en el presente proyecto.

Elementos básicos que conforman el prototipo de video vigilancia.

- Router:
  - TP-LINK TD-W8951ND 150 Mbps Wireless N ADSL2 Modem Router.
- Internet:
  - Ancho de banda: Proveedor de Internet es Pancho Net, 1.22 Mbps de bajada y 0.26 Mbps de subida.
- Cámara IP:
  - SP-FJ01W Pt Wireless IP Camera. Resolución 640x480, 320x240, Supporting multiple network protocols: HTTP/TCP/IP/UDP/STMP/DDNS/FTP, Wireless 802.11 g/b, WEP, WPA, WPA2.

3.2. Diseño del Sistema

- Para el prototipo de video vigilancia se toma en cuenta que existen dos puntos de vista a considerar en la implementación del sistema, el primero es la aplicación web como tal, que permite las funciones de registro, login, y acceso a las cámaras IP, en la que mediante un servicio de TOMCAT se ejecuta la aplicación web y el usuario pueda interactuar desde la web hacia las cámaras a través de un servicio de autentificación proporcionado por la aplicación web. La aplicación consta básicamente de páginas de tipo HTML, así como de códigos fuente java específicamente *servlet.java*. (ver Anexo 1).
- El usuario a través de un navegador web envía una petición HTTP al servidor, a través de un login, el mismo que es autentificado en el servidor ubicado en un hosting con soporte a TOMCAT a través de la utilización de Servlet de

Java, y es después del cual permite obtener las URLs de las cámaras IP que se encuentran trasmitiendo, haciendo efectiva la redirección a las cámaras que se encuentran en una red LAN privada.

- Una vez el usuario autentificado, la aplicación web direcciona las direcciones IP que posee el cliente en su base de datos, y accede a la red interna privada en la que se encuentran las cámaras mediante el port fowarding de las direcciones IP de la red externa a la red interna.
- Por otro lado se tiene el diseño de red, que según la topología puede ser inalámbrica o través de una red LAN con cables. Es decir, existe detrás de la implementación del prototipo un diseño e infraestructura de red, un dominio contratado para el efecto (www.ecuavisor.com), existe un servicio de traducción del nombre del dominio hasta las direcciones IP del cliente, un servicio de autenticación de usuarios.
- Siendo posible que el usuario pueda manipular mediante la aplicación web con las cámaras conectadas, con la posibilidad de generar movimientos horizontales y verticales en tiempo real.

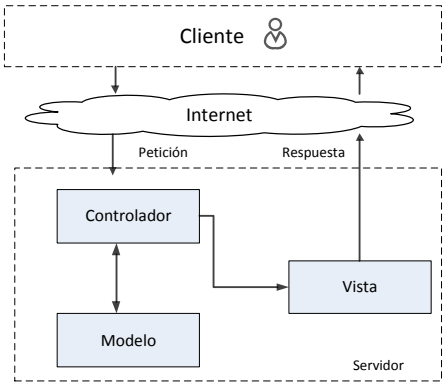
3.3. Arquitectura del programa

3.3.1 Generalidades

El sistema es una aplicación web ejecuta en la plataforma JAVA 1.7.0_03, en la que se usa un servidor web denominado APACHE TOMCAT 7.0.14.0, que interactúa con una base de datos MySql. Con el lenguaje JAVA mediante sus tecnologías y componentes permite la creación de una aplicación web de java denominada *Java Web Application,* lenguaje de programación basado en

etiquetas o marcas que definen la estructura del texto y la forma de desplegar la información.

En la aplicación web cumple con el modelo MVC Model View Controller, en la que se separa los datos de una aplicación, la interfaz de usuario y la lógica de control. El modelo maneja la gestión de la Base de Datos y la lógica del negocio. La vista es una página HTML y código JAVA SCRIPT que permite tener una página web dinámica. El controlador que controla los eventos a través de las peticiones que realiza el usuario al modelo y dentro de la vista. (Ver Ilustración 16)



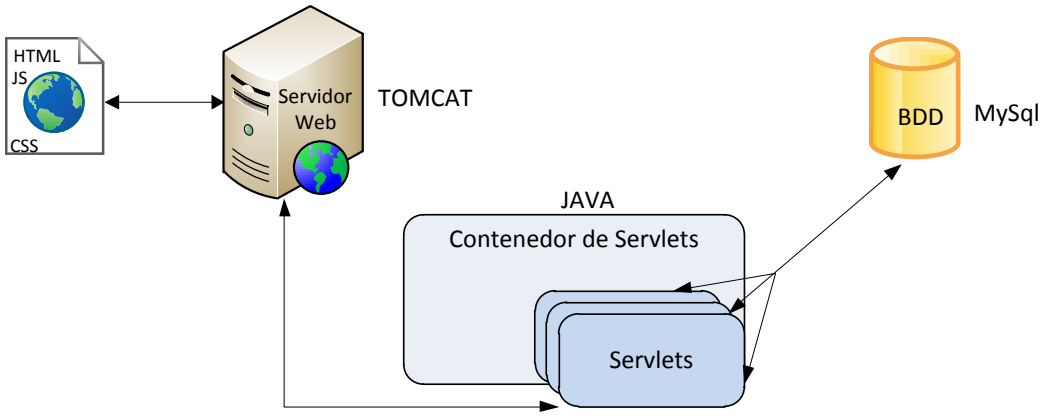
Fuente: http://www.librosweb.es/symfony_1_2/capitulo_2/el_patron_mvc.html.

Ilustración N° 16. Modelo MVC.

Los componentes web utilizados en el sistema de video vigilancia con cámaras IP, son los Servlet, que permite la encapsulación de objetos, además de lograr la interacción con la base de datos y servicios web, se puede usar también tecnologías JSP y JSF en la que la aplicación web para recibir peticiones de clientes y por medio del servidor generar respuestas así como para mejorar la interfaz de usuarios.[19] Lo último no se utiliza para la implementación del prototipo.

[19] Internet página librosweb.es

Se puede graficar la arquitectura la aplicación web del prototipo de video vigilancia como se muestra la Ilustración N° 17.



Fuente: Generación propia.

Ilustración N° 17. Arquitectura de la Aplicación Web Prototipo.

3.3.2 Vista o Interface de Usuario.

En la capa de vista se utiliza una plantilla HTML, divididas en páginas con extensión .html, que permite tener la interfaz gráfica donde se inicia la sesión y se muestran las cámaras IP de video vigilancia. Existe el archivo un registro.html (ver Anexo 1) en la que el usuario o cliente se registra para el acceso al servicio de video vigilancia mediante cámaras IP. Un archivo visor.html (ver Anexo 1) que es la interfaz gráfica en la que se observan la trasmisión de las cámaras IP y pueden ser manipuladas por el usuario, estas páginas HTML se encuentran integradas a la versión de JAVA EE 6. Esta capa permite la interfaz gráfica que el usuario va a manipular para el registro como usuario y de esta manera hacer la petición al servidor de la dirección IP o URLs de las cámaras IPs, teniendo acceso a las cámaras, de esta manera se produce la posibilidad de obtener y brindar

información entre el servidor web y el cliente del usuario que se encuentra autentificado.

Se utiliza JSON (Java Script Object Notation) que es una manera estándar para intercambiar datos entre el navegador web y el servidor. Mediante la librería JSON permite realizar llamadas AJAX con método POST, de esta manera utilizamos AJAX para el envío de datos de forma dinámica.

Dentro de las dependencias que permiten procesar el login de los usuarios, JSON va indicar si se logró o no autentificar el usuario, mediante los mensajes “Nombre de Usuario Incorrecto” o “Contraseña incorrecta”, en el caso de tener éxito en el login de usuario.

Para el desarrollo de la aplicación se cuenta con las siguientes dependencias:

- Commons-beanutils
- Commons-collections
- Commons-logging
- Commons-lang

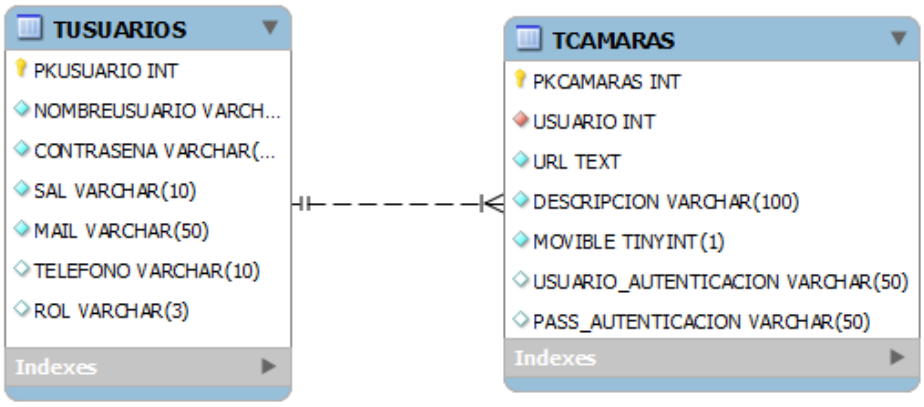
La librería Prototype permite eliminar la complejidad de la programación web del lado del cliente, que posee extensiones para mejorar el entorno de scripts del navegador y proporciona APIs elegantes para las interfaces de AJAX y del DOM Document Object Model.[20]

[20] http://prototypejs.org/

3.3.3 Modelo

La capa Modelo es aquella que contiene los datos de la aplicación, cuando existe una modificación este notifica a la capa de Vista para que se actualice la presentación de los datos. Los Beans que se utilizan son visibles en la persistencia y para el caso del presente prototipo de video vigilancia se utiliza simplemente clases de persistencia que permite crear entidades para el manejo de los datos, la misma que es posible usando eclipselink.[21]

El modelo de la base de datos que utiliza el prototipo de video vigilancia es el presentado en la Ilustración N° 18.



Fuente: Generación propia

Ilustración N° 18. Diagramas ER entidad/relación del modelo de la base de datos.

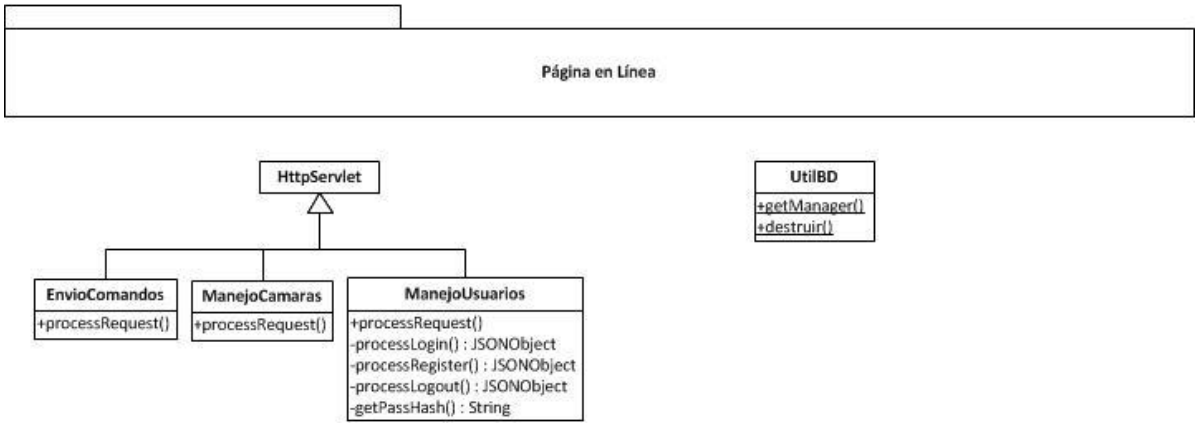
Para la cual se crea un simple modelo de datos que contiene una tabla de Usuarios y otra tabla que contiene la información de las Cámaras, en la relación es de uno a muchos, es decir, un usuario puede tener muchas cámaras.

[21] Harvey M. Deitel, Paul J. pág. 910, Como programar en Java

3.3.4 Controlador

En la arquitectura que se modela en el prototipo de video vigilancia con cámaras IP, el Servlet es el que hace de controlador, permitiendo cumplir con la característica de esta capa que es la gestión del flujo de contenidos tanto en la entrada y salida. Con la ayuda de los Servlet se puede recuperar la información que está en la petición del cliente, dicha información es enviada al servidor web o en otros casos a un EJB o JavaBeans para el proceso de los datos, para luego recuperar los datos e integrar en la respuesta que se enviara al cliente.[22]

La implementación de las clases java Servlet que va a permitir la funcionalidad del sistema de video vigilancia con cámaras IP mediante internet, se presenta en la Ilustración N° 19.



Fuente: Generación propia

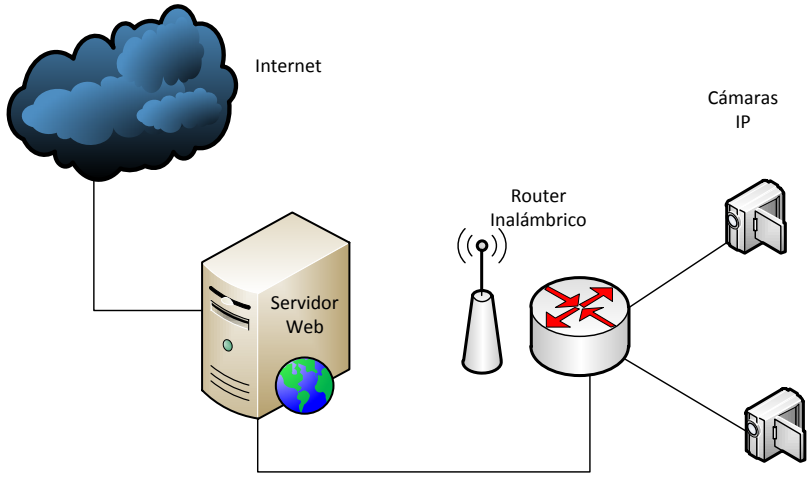
Ilustración N° 19. Diagramas de clases.

3.4 Diseño de la red de las cámaras IP.

En el diseño de red se toman en cuenta aspectos como la ubicación de las cámaras IP, la topología de red, el direccionamiento IP para las los dispositivos, el

[22] Recursos Informáticos J2EE, Benjamín Aumaille, pág. 23.

número de puerto para el enlace de las cámaras IP, para la cual se toma en consideración el servicio de internet, el servicio que ofrece la aplicación web implementada, la configuración del router y de las cámaras IP, según se muestra en la Ilustración N° 20 del diseño lógico de la red.



Fuente: Generación propia

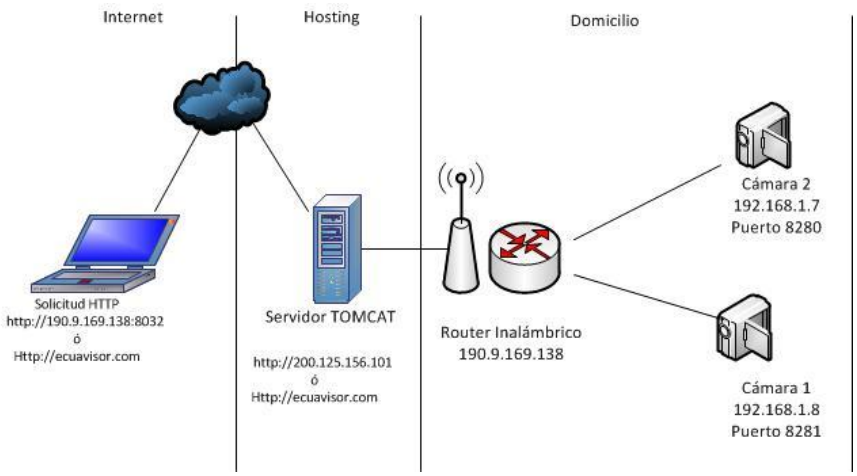
Ilustración N° 20. Diagramas lógico de la red de video vigilancia.

Para la ubicación de las cámaras se considera los puntos más críticos y que mayor vigilancia demanda, como son el ingreso de los domicilios con una cámara IP fija, para exteriores la misma que se coloca en un lugar que cubra la mayor cantidad de espacio para identificar ingreso o salidas del domicilio. La segunda cámara IP está destinada para una vigilancia interna, la misma que va a permitir monitorear puntos críticos del domicilio, como lugares donde se encuentran personas, objetos de mayor valor. Para los otros lugares que se consideren críticos, a criterio del usuario pueden ser colocados de acuerdo a la demanda y disponibilidad de equipos que se adquieren en lo posterior.

Para la topología de red del prototipo de video vigilancia se utiliza dirección IP clase C que fluctúa entre el rango 192.0.0.0 hasta 192.168.255.255 en la que permite tener un numero de Host de 254, con la máscara de red de 255.255.255.0.

El "*puerto"* es aquel que define un servicio, en el caso del prototipo define las solicitudes que recibe el servidor web a través del cliente para acceder a las cámaras IP y le permite procesar los datos entrantes. Un servicio web vía http que se utiliza para la implementación del sistema de video vigilancia se le asigna al puerto 80 de una cámara de red.

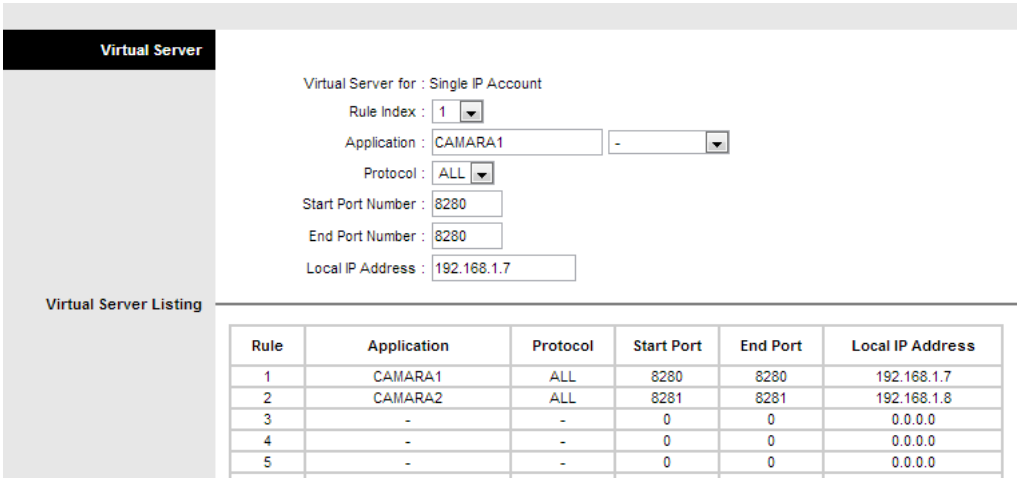
La configuración de las direcciones IPv4, se la realiza manualmente, una dirección IP estática en el software de gestión de la cámara, la máscara de red y la dirección IP del router. O también se la puede realizar de forma automática con el DHCP siempre y cuando se configure la misma IP, que se le conoce como reserva de IP mediante DHCP, ésta última configuración es tomada en cuenta para el presente proyecto. (Ver Ilustración N° 21)



Fuente: Generación propia

Ilustración N° 21. Configuración de IPs en la red de video vigilancia.

Así mismo la configuración del router inalámbrico se presenta en la Ilustración N°22 en la que se puede apreciar que se encuentran reservadas las dos direcciones IP que siempre van a tener las cámaras IP, mediante la creación de reglas en la opción de configuración avanzada del router inalámbrico que provee la empresa que ofrece el servicio de internet. La misma que es escalable para otras cámaras que en lo posterior se deseen activar, en donde se especifica también los puertos que van a servir de túnel para trasmitir las imágenes.



Rule	Application	Protocol	Start Port	End Port	Local IP Address
1	CAMARA1	ALL	8280	8280	192.168.1.7
2	CAMARA2	ALL	8281	8281	192.168.1.8
3	-	-	0	0	0.0.0.0
4	-	-	0	0	0.0.0.0
5	-	-	0	0	0.0.0.0

Fuente: Generación propia

Ilustración N° 22. Configuración de las IP estáticas en el router que poseen las cámaras IP.

La instalación de las cámaras IP está formada por dos lugares críticos determinados por el usuario.

Lugar	Área de cobertura	Equipos existentes	Tipo de cámara
Entrada al Domicilio	Ingreso al domicilio	Ninguno	Fija
Sala	Mayor objetos de valor	Dos Laptop Router D-Link	PT

Fuente: Generación propia

Tabla N° 4. Ubicación de cámaras IP en la red de video vigilancia.

Para el prototipo se toma en cuenta la utilización de las dos cámaras que se mencionan en el la Tabla N°4, la misma que dependiendo de las necesidades del cliente o usuario podría aumentar.

4. DIAGNÓSTICO Y CONSIDERACIONES DEL SISTEMA

En la red interna LAN (Local Área Network), se utiliza direcciones de red clase C, que corresponde a la 192.168.1.0, con mascara 255.255.255.0, es decir que se puede tener un número máximo de hosts de 254, que representan las cámaras IP, pero de acuerdo al ancho de banda se contempla un cálculo posterior para un número real de cámaras IP que se pueden tener en la red.

Para el análisis se considera el uso de internet que realiza el usuario como tal de su servicio de internet, es decir, se toma en cuenta dos escenarios, el uno cuando no existe tráfico o navegación en la red, y la otra cuando existe navegación por internet adicional de la trasmisión de video de las cámaras IP.

Para la implementación del presente prototipo se toma en cuenta, el servicio ofrecido por ISP o proveedor de internet, que para el caso es PANCHO NET, en la que se tiene un plan contratado de 1 Mbps, divididos en 1.22 Mbps de bajada y 0.26 Mbps de subida. Es decir que según el siguiente calculo:

0,26 Mbps o 260 kbps de subida, / 40kbps = 6,5 ≅ 6 cámaras IP que se pueden instalar.

Se cuenta con un dominio denominado *ecuavisor.com* con una IP pública 200.125.156.101, además de contar con un router con IP pública que es la que permite acceder desde la nube hacia la red interna.

El análisis se realiza en horarios de 10H00 a 14H00, que es cuando no se tiene mayor tráfico de red, debido a la ausencia de personas en el inmueble, y en el horario de 18H00 a 22H00 que es cuando el tráfico de red es mayor, debido al

uso del internet por habitantes del inmueble. En donde la medición del tráfico de red se realiza en una semana en días indistintos.

4.1 Tráfico de Red

El propósito del diagnóstico es determinar el tráfico de la red LAN interna, para lo cual con la ayuda de varios software comerciales libres, permite realizar el monitoreo de trafico de las trasmisiones que realizan las cámaras IP. Para lo cual se toma en cuenta un periodo efectivo de 5 días, las 24 horas del día.

Los principales analizadores de tráfico que se consideran para el análisis son los siguientes:

Wireshark es un analizador de protocolos que se utiliza para el análisis y el troubleshooting (solucionador de problemas) en redes. Es un software libre que se ejecuta en la mayoría de sistemas operativos, incluyendo algunas distribuciones LINUX, así como Microsoft Windows. Entre los aspectos importantes de Wireshark está la posibilidad de capturar datos de la red, y obtener estadísticas de los paquetes capturados.[23]

Otra herramienta que se utiliza para el efecto es “Colasoft Capsa Free”, que permite obtener mediciones del ancho de banda de una red. Este software permite monitorear el tráfico de red y el ancho de banda en red, presentando los datos de forma numérica y grafica lo cual es amigable para la presentación de resultados.

El diagnóstico se realiza con las dos cámaras IP dentro de la red LAN interna, en donde el análisis del ancho de banda se realiza mediante el uso del

[23] http://es.wikipedia.org/wiki/Wireshark

internet bajo los protocolos HTTP sobre TCP, es decir se verifica como las imágenes viajan por la red, así como se realiza el análisis del uso del ancho de banda.

Para las mediciones se consideran periodos de 24 horas, poniendo énfasis en horas donde el tráfico del uso de internet es mayor, por ejemplo a partir de las 18H00 que las personas comienzan a utilizar el internet, por otro lado también se considera el tiempo que permanece sin usar el internet y el ancho de banda es totalmente dedicado al monitoreo de las cámaras.

4.2 Análisis con Wireshark

Según el análisis con Wireshark, se estima la conexión a internet que el usuario realiza al solicitar el acceso a las cámaras IP dentro de la red interna LAN, desde la cual se analizó obteniendo los siguientes resultados:

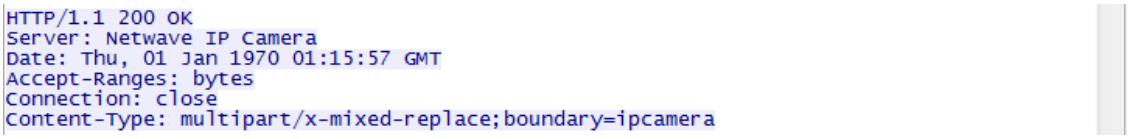
El contenido de los paquetes que son capturados por el software, en la que se inicia con una petición, una respuesta, y una división de imágenes que se van presentando conforme se van trasmitiendo las imágenes en las cámaras IP, así como se muestra a continuación,

En la Ilustración N° 23, se determina la petición GET que se realiza, para la trasmisión de las imágenes a través del host de la cámara IP, en la que existe autentificación básica hecha por el navegador.

Fuente: Generación propia

Ilustración N° 23. Petición que realiza el protocolo HTTP.

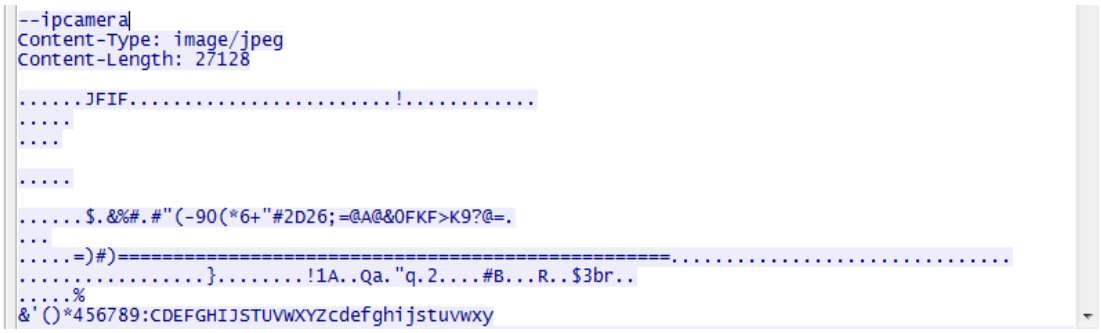
En la Ilustración N° 24, se determina la respuesta hecha a la petición, normalmente se determina con el número 200. Además de determinar el tipo de contenido del paquete que se encuentra viajando por la red, en la que tenemos el tipo “multipart/xmixed-replace;”



Fuente: Generación propia

Ilustración N° 24. Respuesta al navegador.

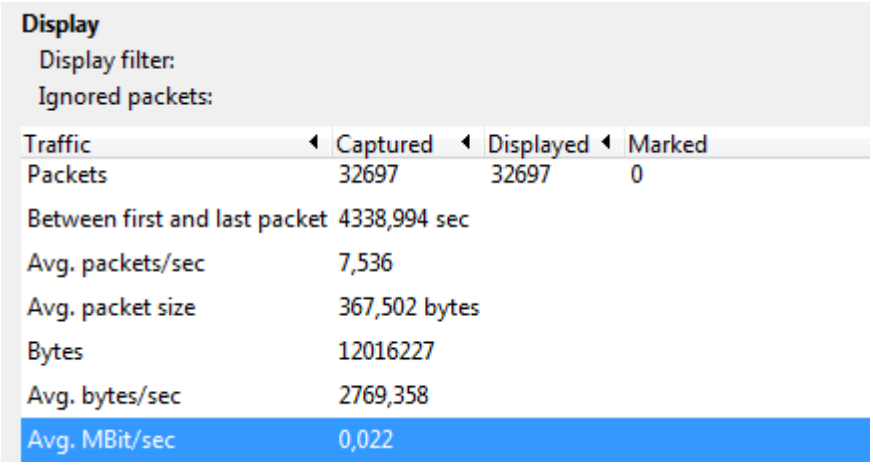
En la Ilustración N° 25, se determina como la imagen se encuentra viajando por la red dentro del paquete de datos, inicia con el boundary=ipcamera, es decir se determina el inicio y fin de las imágenes.



Fuente: Generación propia

Ilustración N° 25. Parte del contenido de las imágenes que viaja por la red.

En la Ilustración N° 26, se determina la cantidad de tráfico capturado por el software, y lo esencial que se puede considerar es el promedio de megabits por segundo utilizados para la trasmisión de imágenes que en este caso es 0,022 MBits por segundo.



Display

Display filter:

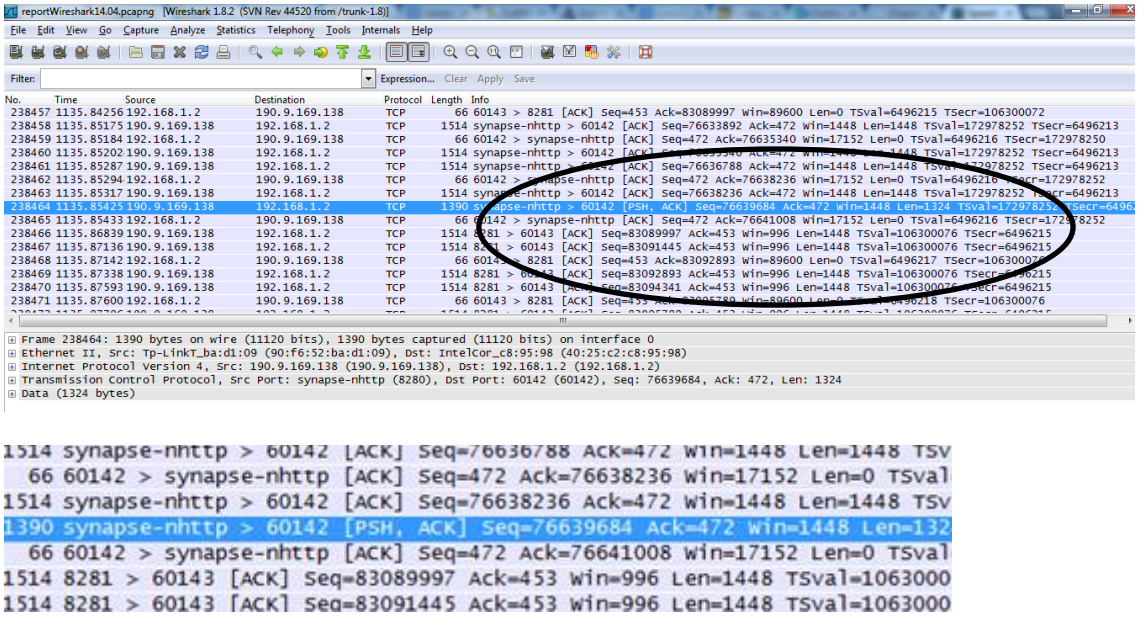
Ignored packets:

Traffic	Captured	Displayed	Marked
Packets	32697	32697	0
Between first and last packet	4338,994 sec		
Avg. packets/sec	7,536		
Avg. packet size	367,502 bytes		
Bytes	12016227		
Avg. bytes/sec	2769,358		
Avg. MBit/sec	0,022		

Fuente: Generación propia

Ilustración N° 26. Resumen del tráfico capturado por Wireshark.

Básicamente en la Ilustración N° 27, se tiene la información de todos los paquetes que viajan por la red, esto es, la hora, la IP origen y destino, el protocolo utilizado, el tamaño del paquete y la información del mismo. Se puede apreciar que la trasmisión de la imagen se inicia con ACK que significa acknowledge o simplemente una advertencia, en la que lo que concierne al prototipo es poder identificar el protocolo PSH o Server push que es el que las cámaras IP implementan para su trasmisión, de esta manera se conoce el momento que inicia la trasmisión de la imagen así como cuando termina su trasmisión.



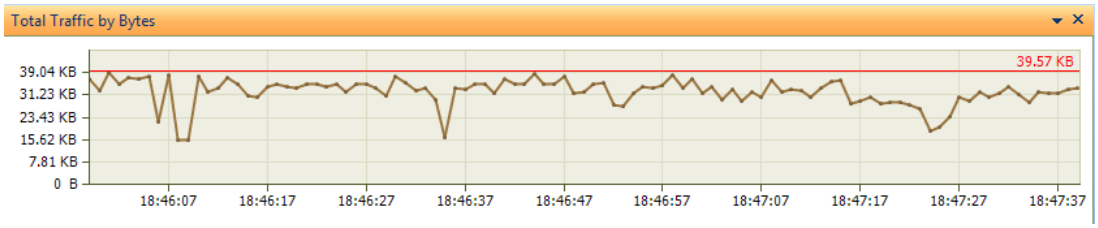
Fuente: Generación propia

Ilustración N° 27. Captura de los paquetes en red con Wireshark.

4.3 Análisis con Colasoft

El ancho de banda que las cámaras usan para transmitir los datos por internet es variable y depende enteramente de la velocidad de subida del plan de internet contratado. Es decir, la máxima velocidad de transmisión de las imágenes de la cámara por internet depende del límite de subida impuesto por el ISP. A continuación ilustraremos esto con dos proveedores: Panchonet y TVCable.

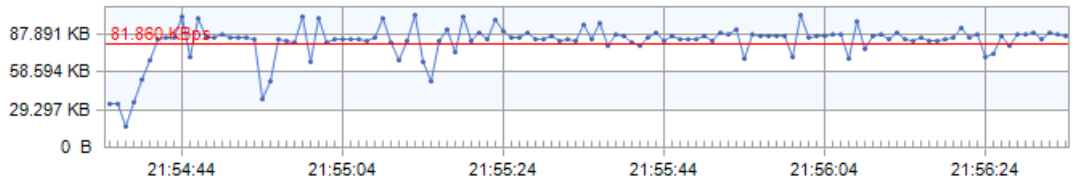
La Ilustración N° 28, es una medición del ancho de banda usado al conectarse al sitio web para observar las cámaras desde internet. Esta cámara se encuentra dentro de la red del cliente Panchonet. Se determina que la cámara transmite la secuencia de imágenes a 39,57 KB.



Fuente: Generación propia

Ilustración N° 28. Total de tráfico utilizado por el uso de internet con las cámaras.

La Ilustración Nº 29 en cambio muestra una cámara de un cliente TVCable. Para este cliente, la velocidad de subida era de aproximadamente 87 Kbps (con un plan de datos contratado de 1,5 Mbps), ancho de banda usado por completo al momento de conectarse al sitio web y revisar la cámara.



Fuente: Generación propia

Ilustración Nº 29. Gráfico de ancho de banda para cliente TVCable.

4.4 Análisis de seguridad de la red LAN

En cuanto a este acápite, se tiene las siguientes consideraciones acerca de las seguridades que ofrece el servicio de monitoreo de cámaras IP a través del prototipo:

- Los usuarios son habilitados mediante un administrador que recoge los datos, los almacena y lo gestiona en la base de datos, lo cual hace que no existan usuarios independientes y maliciosos o falsos.

- Con implementación de HTTPs se brinda una capa de seguridad de manera que no sean visibles las imágenes ni las contraseñas en la WAN.

- El router posee un DHCP para la red WLAN, el cual es configurable y nos permite asignar una IP fija para cada cámara cada vez que se enciende.

- Con la instalación de una VPN se puede encriptar la información que viaja en la red, haciendo más segura la interconexión de paquetes dentro de la red.

- Los puertos que se encuentran habilitados en la configuración del router deben permanecer configurados para asegurar que la conexión a las cámaras IP sea exitosa. Nótese que esto no es requerido si se usa una VPN.

- El programa no debería enviar la IP del router del cliente al navegador. Las imágenes deberían ser procesadas en la aplicación web y enviadas al navegador desde el mismo dominio.

4.5 Análisis de costos

Para caso del prototipo de video vigilancia se toman en cuenta aspectos como el precio de las cámaras IP, el costo que demanda el servicio de internet mensual, el costo de IP pública para el caso de acceder remotamente, así como temas de instalación y mantenimiento, en el caso del prototipo en mención posee servicio de internet mensualmente, que provee Pancho Net a través obviamente de CNT por la línea ADSL.

La cámara IP está valorada en el mercado ecuatoriano desde los 90 dólares hasta los 500 dólares, dependiendo de las funcionalidades de las cámaras IP. Para el caso del prototipo se cuenta con una cámara móvil valorada en 95 dólares americanos que de ser el caso sería el precio referencial para la importación, y otra cámara fija valorada en el mismo precio de 95 dólares americanos.

El servicio de internet depende de la velocidad de comunicación que se contrate, actualmente el mercado ofrece entre 1 Mbps hasta 8 Mbps, para el caso del prototipo que se encuentra en objeto de estudio es 150kbps, divididos en un 1.08 Mbps de bajada y de subida que interesa es de 0,26 Mbps o 260 Kbps.

Para el caso de instalación y mantenimiento de las cámaras se considera un costo de 200 dólares anuales, tomando en cuenta que las empresas normalmente cubren costos de operación y movilización, además se pude incluir un costo por el asesoramiento periódico que se le brinde para del funcionamiento de las cámaras de video vigilancia. Otro caso muy diferente es el caso de la grabación de video a través de un NVR por sus siglas Network Video Record, que en el presente trabajo no se encuentra contemplado. Por lo que para nuestros fines de comparación hemos tomado un punto referencial de $95 dólares, por conceptos de revisión de configuración e imprevistos, dejando en claro que estos imprevistos pueden darse lugar en el trascurso del año de servicio de video vigilancia.

El router y/o, modem que se encuentra en el domicilio no se toma en cuenta para efectos del prototipo ya que se considera que los usuarios poseen un contrato de servicio de internet en el que viene incorporado estos dispositivos al

momento de la contratación del servicio. Además no se toma en cuenta las computadoras portátiles o de escritorio y celulares que son utilizados para el monitoreo porque se asume que la mayoría de usuarios ya lo poseen.

Cantidad	Producto	Valor Unitario	Valor Total
2	Cámara Wireless IP	$ 95.00	$ 190.00
1 año de internet	Servicio de Internet	$ 21.00	$ 252.00
1 año	Instalación y Mantenimiento	$ 95.00	$ 95.00
1 año	Extra	$ 20 - $50.00	$ 50.00
		Total	**$ 587**

Fuente: Generación propia

Tabla N° 5. Costo referencial del prototipo de video vigilancia para el usuario.

Cantidad	Producto	Valor Unitario	Valor Total
2	Cámara Wireless IP	$ 235.00	$ 470.00
1 año de internet	Servicio de Internet	$ 21.00	$ 252.00
1	IP Pública	$ 12	$ 144.00
1 año	Instalación y Mantenimiento	$ 150.00	$ 150.00
1	Extra	$ 20 - $50.00	$ 50.00
		Total	**$ 1066**

Fuente: Generación propia

Tabla N° 6. Costo referencial de video vigilancia para el usuario ofrecido por empresas en el mercado.

Cantidad	Producto	Valor Unitario	Valor Total
4	Cámara LAN	$ 235.00	736.52
1 año de internet	Servicio de Internet	$ 21.00	$ 252.00
1 año	IP Pública	$ 12	$ 144.00
1 año	Extra	$ 20 - $50.00	$ 50.00
		Total	**$ 1182,52**

Fuente: Generación propia

Tabla N° 7. Costo referencial de video vigilancia para el usuario ofrecido por oferta de kit en empresas en el mercado.

En la Tabla N°5 se muestra la cantidad de costo que se invierte para ofrecer el servicio de video vigilancia ofrecido por ecuavisor.com (Prototipo de video vigilancia del presente proyecto). Se tiene un costo referencial de 600 dólares americanos frente a los 1100 y 1200 dólares americanos con los que se debe pagar en empresas que ofertan el servicio de video vigilancia. Es importante aclarar que según la Tabla N°7 el costo total constituye en una oferta de un plan de cuatro cámaras, en la que la instalación en su conjunto es cableada, y en la que se pide por parte de este tipo de empresa la necesidad de incorporar el costo de la IP publica para poder acceder desde un sitio remoto fuera de la red interna del sistema de video vigilancia contratado.

5. CONCLUSIONES

Desde el punto de vista del usuario

- El prototipo está pensado para funcionar dentro de una red inalámbrica debido a que disminuye los costos de implementación (no necesita poner cableado de red) y permite que el usuario ubique sus cámaras donde le resulte más conveniente. Sin embargo, las cámaras no poseen batería, por lo que la ubicación de las cámaras está limitada a los lugares donde existan enchufes de corriente eléctrica cercana.

- El prototipo de video vigilancia con cámaras IP usa el concepto de video vigilancia doméstica, lo que significa que es de bajo costo (ver Tabla N°5), no requiere configuración por parte del usuario, y existe un servicio de soporte técnico.

- Para ser parte de este servicio no existen muchos requerimientos. Se necesita:

    - Un documento con sus datos personales (incluyendo dirección y teléfonos) para el registro dentro del sistema, así como para verificación. También debe incluir cuántas cámaras desea tener.

    - Una conexión a internet de cualquier proveedor. De acuerdo a las ilustraciones 28 y 29, vemos que las cámaras se ajustan al ancho de banda de subida impuesto por el proveedor, siendo el valor de 37 Kbps el mínimo recomendado. Nótese que al momento de conectarse a las cámaras, el ancho de banda de subida se ocupa

por completo, mas no el ancho de banda de bajada. Esto permite que los usuarios de la casa puedan seguir navegando en internet con un mínimo impacto en la velocidad de conexión (debido a que para visitar sitios web se usa el ancho de banda de bajada).

- En cuanto a seguridades, el prototipo no presenta vulnerabilidades dentro de la red en sí ya que solo expone al internet las cámaras mediante HTTP. No se permite el ingreso a ningún otro servicio. Además se configura el ruteador casero para que solo permita conexiones desde nuestro servidor (ecuavisor.com) protegiendo así la privacidad de los usuarios.

- Como medida extra de seguridad, los usuarios no pueden registrarse por sí mismos en la página. El ingreso de usuarios y cámaras está a cargo del administrador del sistema. Él les asigna un usuario para que puedan usar el servicio.

Desde el punto de vista del servicio ofrecido.

- La situación actual de tecnología de video vigilancia por internet en nuestro país, se encuentra aun sin acceso a la mayoría de las personas, (31,4 de la población de Ecuador ha utilizado internet en el año 2011), debido a que resulta en la mayoría de casos costoso, inseguro o simplemente su implementación resulta un tanto compleja para los usuarios.

- Desde el internet, el ancho de banda que consume cada cámara es de aproximadamente 40 kbps a una resolución de 320x240 pixeles. Esto permite que las cámaras puedan ser vistas por personas con conexión ADSL y desde celulares con plan de datos.

- El prototipo, al ofrecer un servicio web, está sujeto a las limitaciones de los navegadores. Debido a que usa el tipo MIME multipart-mixed-replace, se requiere un navegador que soporte este formato. Chrome, Firefox, Opera y sus equivalentes para teléfonos móviles lo soportan. Internet Explorer implementa este formato desde la versión 10[14].

- Para mejorar la seguridad de la transmisión de datos sobre internet, es posible poner un certificado HTTPS en nuestro servidor Tomcat.

- El prototipo es escalable en el sentido de que se puede ingresar un número arbitrario de usuarios. Es posible registrar nuevos usuarios o agregarles cámaras en cualquier momento. El tráfico de video no pasa por el servidor, por lo tanto no existe un límite en la cantidad de datos que se pueden transmitir desde la cámara hacia el navegador del usuario.

6. RECOMENDACIONES

- Se recomienda pruebas adicionales para la red inalámbrica para determinar mejor cobertura y recepción de la señal en los sitios considerados como críticos.

- Se recomienda además el uso de UPS para las cámaras para momentos de cortes de corriente eléctrica, de ser el caso de poseer un gran número de cámaras IP.

- Se recomienda ubicar las cámaras en sitios donde no sea de fácil acceso y no permitan la manipulación por personas internas como externas.

- Se recomienda un análisis más profundo en cuanto a las funcionalidades que ofrece las cámaras IP como son, alarmas, envío de mensajería, correo o manejo de eventos con movimientos.

- De ser el caso de aumentar la resolución de imagen de las cámaras se puede lograr una imagen aún más clara con 640x480 pixeles, pero se recomienda aumentar la velocidad de trasmisión contratados con el proveedor de servicios de internet.

BIBLIOGRAFIA

1. García Mata. (2012). *Video vigilancia: CCTV usando videos IP*. España: Vértice.
2. Stallings, William. (2004). *Comunicaciones y redes de computadores Séptima Edición*. España: Pearson.
3. Castillo, Juan Mármol (2010). *Persistencia de Objetos. JDO, Solución Java*: Recuperado 2 de enero 2013, de sitio web, http://dis.um.es/~jmolina/Persistencia%20de%20Objeto%20JDO.pdf.
4. Harvey M. Deitel, Paul J Deitel. (2004). *Como programar en Java Quinta Edición*. México: Pearson Educación.
5. Aumaille, Benjamín. (2002). *Recursos Informáticos J2EE- Desarrollo de aplicaciones Web*. Barcelona, España: Ediciones ENI.
6. Alfonso García-Cervigón Hurtado María del Pilar Alegre Ramos (2011). *Seguridad Informática.* Madrid, España.: Paraninfo.
7. Romero Laguillo, Luis Fernando(1997). *Publica en Internet*: *Guía Práctica Creación de Documentos HTML.* España: Publicaciones de la Universidad de Cantabria.
8. Prototype.org (2012). Recuperado 3 de enero 2013, de sitio web http://prototypejs.org/
9. Libros Web.es (2013). Recuperado 4 de noviembre 2012 de sitio web http://www.librosweb.es/symfony_1_2/capitulo_2/el_patron_mvc.html
10. Project Lombok (2009). Recuperado 4 de diciembre de 2012, de sitio web http://projectlombok.org/setup/netbeans.html

11. Stackoverflow.com (2013). Recuperado 2 de enero 2013, de sitio web http://stackoverflow.com/
12. Developer.com (2013). Recuperado 3 de enero 2013, de sitio web http://www.developer.com/java/data/article.php/3417381/Using-JDBC-with-MySQL-Getting-Started.htm
13. MySQL.com (2013). Recuperado 3 de febrero 2013, de sitio web http://www.mysql.com/
14. Apache Commons (2013). Recuperado 3 de febrero 2013, de sitio web http://commons.apache.org/codec/
15. XMLHttpRequest Enhancements (2013). Recuperado 3 de mayo de 2013, sitio web http://msdn.microsoft.com/en-us/library/ie/hh673569%28v=vs.85%29.aspx#Download_streaming_support

Anexo 1

Código Fuente del prototipo de video vigilancia

EnvioComandos.java

```
package com.marcelo;
import com.marcelo.persistencia.Camara;
import java.io.IOException;
import java.io.PrintWriter;
import java.net.URL;
import java.util.HashMap;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceException;
import javax.persistence.Query;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import net.sf.json.JSONObject;
import org.apache.commons.lang.StringUtils;
import org.apache.http.auth.AuthScope;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.BasicResponseHandler;
import org.apache.http.impl.client.DefaultHttpClient;

/**
 * Clase que envía comandos a las cámaras.
 * @author marcelo
 */
public class EnvioComandos extends HttpServlet {

    private static HashMap<String, String> COMANDOS = new HashMap<String,
String>(4);

    static {
        COMANDOS.put("ARRIBA", "0");
        COMANDOS.put("PARAR", "1");
        COMANDOS.put("ABAJO", "2");
        COMANDOS.put("IZQUIERDA", "4");
        COMANDOS.put("DERECHA", "6");
    }
```

```
   /**
    * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
methods.
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
   protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
          throws ServletException, IOException {
       response.setContentType("application/json; charset=UTF-8");
       PrintWriter out = response.getWriter();
       boolean debug = "true".equals(StringUtils.defaultIfBlank(
             getServletContext().getInitParameter("debug"), "false"));

       JSONObject respuesta = new JSONObject();

       //Comprobar que se haya iniciado sesión
       HttpSession sesion = request.getSession(false);
       if (sesion == null || sesion.getAttribute("autenticado") == null ||
             (Boolean) sesion.getAttribute("autenticado") == false) {
          respuesta.element("error", true);
          respuesta.element("mensaje", "No ha iniciado sesión.");
          out.write(respuesta.toString());
          return;
       }

       EntityManager manager = UtilBD.getManager();
       Query consulta = manager.createQuery("SELECT OBJECT(camara) FROM Camara
camara "
          + "WHERE camara.pkcamara = :id");
       try {
          consulta.setParameter("id", Integer.parseInt(request.getParameter("uid")));
       } catch (NumberFormatException ex) {
          respuesta.element("error", true);
          respuesta.element("mensaje", "Id de cámara no válido!");
          out.write(respuesta.toString());
          return;
       }

       Camara cam = null;

       try {
          cam = (Camara) consulta.getSingleResult();
       } catch (PersistenceException ex) {
          respuesta.element("error", true);
          respuesta.element("mensaje", ex.getMessage());
```

```
            return;
        }

        if (cam == null) {
            respuesta.element("error", true);
            respuesta.element("mensaje", "No se encontró cámara con id = "
                    + String.valueOf(request.getParameter("uid")));
            return;
        }

        //Armar la nueva URL a la que se va a enviar el comando
        URL url = new URL(cam.getUrl());
        URL baseUrl = new URL(url.getProtocol(), url.getHost(), url.getPort(), "/");
        String cmdURL = baseUrl.toString() + "decoder_control.cgi?command=";
        String comando = request.getParameter("cmd");

        if (StringUtils.isBlank(comando) || StringUtils.isBlank(COMANDOS.get(comando)))
{
            respuesta.element("error", true);
            respuesta.element("mensaje", "Comando no reconocido: " +
String.valueOf(comando));
            return;
        }

        cmdURL += COMANDOS.get(comando);
        if (debug) {
            System.out.println("URL comando: " + cmdURL);
        }

        //Iniciar la conexión con la cámara y enviarle el comando solicitado.
        DefaultHttpClient client = new DefaultHttpClient();

        try {
            HttpPost httppost = new HttpPost(cmdURL);
            client.getCredentialsProvider().setCredentials(AuthScope.ANY,
                    new UsernamePasswordCredentials(cam.getUsuarioAutenticacion(),
                    cam.getPassAutenticacion()));
            ResponseHandler<String> handler = new BasicResponseHandler();
            String res = client.execute(httppost, handler);

            if (debug) {
                System.out.println("Respuesta de la cámara: " + String.valueOf(res));
            }
        } catch (Exception ex) {
            System.out.println("Error al enviar petición a cámara! " +
ex.getLocalizedMessage());
            if (debug) ex.printStackTrace();
        } finally {
            client.getConnectionManager().shutdown();
```

```
        }

        try {
            out.write(respuesta.toString());
        } finally {
            out.close();
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign
on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "URLs de cámaras";
    }// </editor-fold>
}
```

ManejoCamaras.java

```
package com.marcelo;
import com.marcelo.persistencia.Camara;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.Query;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import net.sf.json.JSONArray;
import net.sf.json.JSONObject;

/**
 * Clase que se encarga de atender la petición de cámaras.
 * @author marcelo
 */
public class ManejoCamaras extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        response.setContentType("application/json; charset=UTF-8");
        PrintWriter out = response.getWriter();

        JSONObject respuesta = new JSONObject();

        //Comprobar que se haya iniciado sesión
        HttpSession sesion = request.getSession(false);
        if (sesion == null || sesion.getAttribute("autenticado") == null ||
                (Boolean) sesion.getAttribute("autenticado") == false) {
            respuesta.element("error", true);
            respuesta.element("mensaje", "Para poder ver esta página, debe <a
href=\"index.html\">iniciar sesión</a>.");
            out.write(respuesta.toString());
            return;
```

```
        }

        EntityManager manager = UtilBD.getManager();
        Query consulta = manager.createQuery("SELECT OBJECT(camara) FROM Camara
camara "
            + "WHERE camara.usuario = :usr");
        consulta.setParameter("usr", sesion.getAttribute("uid"));
        List<Camara> registros = (List<Camara>) consulta.getResultList();

        if (registros.isEmpty()) {
            respuesta.element("error", true);
            respuesta.element("mensaje", "No tiene cámaras registradas.");
            out.write(respuesta.toString());
            return;
        } else {
            JSONArray arreglo = new JSONArray();
            for (Camara camara : registros) {
                JSONObject elemento = new JSONObject();
                elemento.element("url", camara.getUrl());
                elemento.element("authorization", camara.obtenerCredencialesAuthBasic());
                elemento.element("descripcion", camara.getDescripcion());
                elemento.element("movible", camara.isMovible());
                elemento.element("uid", camara.getPkcamara());
                arreglo.add(elemento);
            }

            respuesta.element("camaras", arreglo);
        }

        try {
            out.write(respuesta.toString());
        } finally {
            out.close();
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign
on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
```

```
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "URLs de cámaras";
    }// </editor-fold>
}
```

ManejoUsuarios.java

```
package com.marcelo;
import com.marcelo.persistencia.Usuario;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.Query;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import net.sf.json.JSONObject;
import org.apache.commons.codec.digest.DigestUtils;
import org.apache.commons.lang.StringEscapeUtils;
import org.apache.commons.lang.StringUtils;

/**
 * Clase que se encarga de login y de registro de usuarios
 * @author marcelo
 */
public class ManejoUsuarios extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        response.setCharacterEncoding("utf-8");
        response.setContentType("application/json");

        String tipo = StringUtils.defaultString(request.getParameter("tipo"), "login");
        JSONObject respuesta = new JSONObject();

        if ("login".equals(tipo)) {
            respuesta = processLogin(request);
        } else if ("logout".equals(tipo)) {
            respuesta = processLogout(request);
        } else if ("registro".equals(tipo)) {
            respuesta = processRegister(request);
```

```
        }

        PrintWriter out = response.getWriter();
        try {
            out.write(respuesta.toString());
        } catch (Throwable t) {
            t.printStackTrace();
        } finally {
            out.close();
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign
on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Servlet de login";
    }// </editor-fold>
```

```
@Override
public void destroy() {
   UtilBD.destruir();
}

public JSONObject processLogin(HttpServletRequest request) {
   boolean debug = "true".equals(StringUtils.defaultIfBlank(
         getServletContext().getInitParameter("debug"), "false"));
   //Si ya existe una sesión activa, destruirla.
   HttpSession sesion = request.getSession(false);
   if (sesion != null) {
      sesion.invalidate();
   }

   JSONObject respuesta = new JSONObject();
   String usuario = request.getParameter("nombre");
   String pass = request.getParameter("pass");

   EntityManager manager = UtilBD.getManager();
   Query consulta = manager.createQuery("SELECT OBJECT(u) FROM Usuario u "
      + "WHERE u.nombreUsuario = :usr");
   consulta.setParameter("usr", usuario);
   List<Usuario> registros = (List<Usuario>) consulta.getResultList();

   boolean exito = false; //Indica si fue un login exitoso.

   if (!registros.isEmpty()) {
      Usuario usr = registros.get(0);
      String sal = usr.getSal();
      String hashUsuario = getPassHash(sal, pass);

      if (hashUsuario.equals(usr.getContrasena())) {
         exito = true;

         sesion = request.getSession(true);
         sesion.setAttribute("autenticado", true);
         sesion.setAttribute("uname", usr.getNombreUsuario());
         sesion.setAttribute("uid", usr.getPkusuario());
      }

      if (debug) {
         System.out.println("Encontrado registro de usuario: hashUsuario: "
               + hashUsuario + "\nHashBD: " + usr.getContrasena());
      }
   }

   respuesta.element("exito", exito);

   return respuesta;
```

```
    }

    private JSONObject processRegister(HttpServletRequest request) {
        JSONObject respuesta = new JSONObject();

        String usuario = StringEscapeUtils.escapeSql(request.getParameter("nombre"));
        String pass = StringEscapeUtils.escapeSql(request.getParameter("pass"));
        String correo = StringEscapeUtils.escapeSql(request.getParameter("correo"));
        String telf = StringEscapeUtils.escapeSql(request.getParameter("telefono"));
        String rol = StringEscapeUtils.escapeSql(request.getParameter("rol"));

        EntityManager manager = UtilBD.getManager();
        Query consulta = manager.createQuery("SELECT OBJECT(u) FROM Usuario u "
            + "WHERE u.nombreUsuario = :usr");
        consulta.setParameter("usr", usuario);
        List<Usuario> usuarios = consulta.getResultList();

        if (!usuarios.isEmpty()) {
            //Nombre de usuario ya existe.
            respuesta.element("error", true);
            respuesta.element("mensaje", "El nombre de usuario ya existe.");
            return respuesta;
        }

        //Crear sal para el usuario
        String sal = DigestUtils.sha256Hex(Double.toString(Math.random())).substring(0,
10);
        pass = getPassHash(sal, pass);
        Usuario usr = new Usuario();
        usr.setContrasena(pass);
        usr.setMail(correo);
        usr.setNombreUsuario(usuario);
        usr.setSal(sal);
        usr.setTelefono(telf);

        if (!rol.matches("USR|ADM")) {
            respuesta.element("error", true);
            respuesta.element("mensaje", "Rol incorrecto!");
            return respuesta;
        }

        usr.setRol(rol);
        manager.persist(usr);

        return respuesta;
    }

    private String getPassHash(String sal, String contrasena) {
        String concatenada = sal + ":" + contrasena;
```

```
        return DigestUtils.sha256Hex(concatenada);
    }

    private JSONObject processLogout(HttpServletRequest req) {
        HttpSession sesion = req.getSession();
        if (sesion != null) {
            sesion.setAttribute("autenticado", false);
            sesion.invalidate();
        }

        JSONObject res = new JSONObject();
        res.element("url", "index.html");

        return res;
    }
}
```

Principal.java

```
package com.marcelo;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author marcelo
 */
public class Principal extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        response.setContentType("application/javascript; charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.write("{\"url\":
\"http://192.168.16.1:8280/videostream.cgi?user=admin&pwd=&resolution=8&rate=0\"}")
;

        } finally {
            out.close();
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign
on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
```

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

UtilBD.java

```
package com.marcelo;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 * Manejador de conexiones BD
 * @author marcelo
 */
public class UtilBD {
   private static EntityManagerFactory factory;

   static {
      try {
         Class.forName("com.mysql.jdbc.Driver");
      } catch (ClassNotFoundException ex) {
         System.out.println("No se encontró dirver mysql!");
      }

      factory = Persistence.createEntityManagerFactory("CamarasPU");
   }

   private UtilBD() {

   }

   public static EntityManager getManager() {
      return factory.createEntityManager();
   }

   public static void destruir() {
      factory.close();
   }
}
```

persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
  <persistence-unit name="CamarasPU" transaction-type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
    <class>com.marcelo.persistencia.Camara</class>
    <class>com.marcelo.persistencia.Usuario</class>
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <shared-cache-mode>NONE</shared-cache-mode>
    <properties>
      <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/camaras"/>
      <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
      <property name="javax.persistence.jdbc.user" value="root"/>
      <property name="javax.persistence.jdbc.password" value="rober710"/>
      <property name="eclipselink.logging.level" value="FINE"/>
    </properties>
  </persistence-unit>
</persistence>
```

Camara.java

```
package com.marcelo.persistencia;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import javax.persistence.*;
import lombok.*;
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.lang.StringUtils;
import org.eclipse.persistence.annotations.ConversionValue;
import org.eclipse.persistence.annotations.ObjectTypeConverter;

@Data
@Entity
@Table(name = "TCAMARAS")
@NoArgsConstructor
@AllArgsConstructor
public class Camara implements Serializable {
   @Id
   @Column(name = "PKCAMARAS")
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   int pkcamara;

   @Column(name = "USUARIO" )
   int usuario;

   @Column(name = "URL")
   String url;

   @Column(name = "DESCRIPCION")
   String descripcion;

   @Column(name = "MOVIBLE")
   @ObjectTypeConverter(
      name = "convertirBoolean",
      dataType = byte.class,
      objectType = boolean.class,
      conversionValues = {
         @ConversionValue(dataValue = "0", objectValue = "false"),
         @ConversionValue(dataValue = "1", objectValue = "true")
      }
   )
   boolean movible;

   @Column(name = "USUARIO_AUTENTICACION")
   String usuarioAutenticacion;

   @Column(name = "PASS_AUTENTICACION")
```

```
    String passAutenticacion;

    public String obtenerCredencialesAuthBasic() {
        String base64 = StringUtils.defaultString(usuarioAutenticacion) + ":"
                + StringUtils.defaultString(passAutenticacion);

        try {
            base64 = Base64.encodeBase64String(base64.getBytes("UTF-8"));
        } catch (UnsupportedEncodingException ex) {
            base64 = Base64.encodeBase64String(base64.getBytes());
        }

        return base64;
    }

}
```

Usuario.java

```
package com.marcelo.persistencia;
import java.io.Serializable;
import javax.persistence.*;
import lombok.*;

@Data
@Entity
@Table(name = "TUSUARIOS")
@NoArgsConstructor
@AllArgsConstructor
public class Usuario implements Serializable {

   @Id
   @Column(name = "PKUSUARIO")
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   int pkusuario;

   @Column(name = "NOMBREUSUARIO")
   String nombreUsuario;

   @Column(name = "CONTRASENA")
   String contrasena;

   @Column(name = "SAL")
   String sal;

   @Column(name = "MAIL")
   String mail;

   @Column(name = "ROL")
   String rol;

   @Column(name = "TELEFONO")
   String telefono;
}
```

camaras.js

```
/**
 * Clase que maneja la página de cámaras
 */

var Camaras = {

   init: function() {
      new Ajax.Request('urls', {
         onSuccess: function(response) {
            var obj = response.responseJSON;
            var divMensaje = $$('.error-usuario')[0];
            var spanMensaje = divMensaje && divMensaje.select('span')[0];

            if (!divMensaje) {
               divMensaje = new Element('div', {
                  'class': 'error-usuario'
               });

               spanMensaje = new Element('span');
               divMensaje.appendChild(spanMensaje);
            }

            if (!obj) {
               spanMensaje.update('Error al obtener datos del servidor.');
               divMensaje.show();
               return;
            } else if (obj && obj.error) {
               spanMensaje.update(obj.mensaje || 'Error al obtener datos del servidor.');
               divMensaje.show();
               return;
            }

            //Crear las camaras
            var contenedor = $('contenedor-camaras');
            var camaras = Object.isArray(obj.camaras) && obj.camaras || [];

            for (var i = 0; i < camaras.length; i++) {
               var cam = Camaras.crearCamara(camaras[i]);
               contenedor.appendChild(cam);
            }
         },

         onFailure: function() {
            var divMensaje = $$('.error-usuario')[0];
            var spanMensaje = divMensaje && divMensaje.select('span')[0];
```

```
            if (!divMensaje) {
                divMensaje = new Element('div', {
                    'class': 'error-usuario'
                });

                spanMensaje = new Element('span');
                divMensaje.appendChild(spanMensaje);
            }

            spanMensaje.update('Error al obtener datos del servidor.');
            divMensaje.show();
        }
    });
},

/**
 * Crea un Element que representa un div con la imagen de la cámara y los controles.
 * @param {Object} camara Objeto con los parámetros de la cámara.
 * @returns {Element} Div con la cámara.
 */
crearCamara: function(camara) {
    var wrapper = new Element('div' , {
        'class': 'camara'
    });

    var img = new Element('img', {
        src: camara.url
    });
    wrapper.appendChild(img);

    var descripcion = new Element('p');
    descripcion.update(camara.descripcion);
    wrapper.appendChild(descripcion);

    if (camara.movible) {
        //TODO: Definir url para controles
        var controles = Camaras.crearControles(camara.uid);
        wrapper.appendChild(controles);
        wrapper.controles = controles;

        wrapper.on('mouseover', function(e) {
            this.controles.show();
        });

        wrapper.on('mouseout', function(e) {
            this.controles.hide();
        });
    }
```

```
    return wrapper;
},

/**
 * Crea un Element que representa una tabla con los controles para la cámara indicada.
 * @param {int} uid Id de la cámara a la que se enviarán los comandos de movimiento.
 */
crearControles: function(uid) {
    var tabla = new Element('table', {
        'class': 'controles',
        'border': '0',
        'style': 'display: none;'
    });

    var tbody = new Element('tbody');

    //Función manejadora de eventos
    var manejador = function(id, cmd) {
        new Ajax.Request('comando', {
            method: 'post',
            parameters: { "uid": id, "cmd": cmd }
        });
    };

    var tr = new Element('tr');
    tr.appendChild(new Element('td'));
    var td = new Element('td');
    var img = new Element('img', {
        src: 'img/arriba.png',
        "class": 'ctl'
    });
    img.on('mousedown', manejador.curry(uid, 'ARRIBA'));
    img.on('mouseup', manejador.curry(uid, 'PARAR'));
    td.appendChild(img);
    tr.appendChild(td);
    tr.appendChild(new Element('td'));
    tbody.appendChild(tr);

    //Segunda fila
    tr = new Element('tr');
    td = new Element('td');
    img = new Element('img', {
        src: 'img/izquierda.png',
        "class": 'ctl'
    });
    img.on('mousedown', manejador.curry(uid, 'IZQUIERDA'));
    img.on('mouseup', manejador.curry(uid, 'PARAR'));
    td.appendChild(img);
    tr.appendChild(td);
```

```
        td = new Element('td');
        td.appendChild(new Element('img', { src: 'img/centro.png' }));
        tr.appendChild(td);
        td = new Element('td');
        img = new Element('img', {
            src: 'img/derecha.png',
            "class": 'ctl'
        });
        img.on('mousedown', manejador.curry(uid, 'DERECHA'));
        img.on('mouseup', manejador.curry(uid, 'PARAR'));
        td.appendChild(img);
        tr.appendChild(td);
        tbody.appendChild(tr);

        //Última fila
        tr = new Element('tr');
        tr.appendChild(new Element('td'));
        td = new Element('td');
        img = new Element('img', {
            src: 'img/abajo.png',
            "class": 'ctl'
        });
        img.on('mousedown', manejador.curry(uid, 'ABAJO'));
        img.on('mouseup', manejador.curry(uid, 'PARAR'));
        td.appendChild(img);
        tr.appendChild(td);
        tr.appendChild(new Element('td'));
        tbody.appendChild(tr);

        tabla.appendChild(tbody);
        return tabla;
    }
};

//Ejecutar el método init al cargar página.
$(document).on('dom:loaded', function(e) {
    Camaras.init();
});
```

login.js

```
document.observe('dom:loaded', function(e){
   $('enviar').on('click', function(e) {
      var nombre = $('nombre').value;
      var pass = $('pass').value;
      var divMensaje = $('mensaje-error');

      if (nombre == '') {
         divMensaje.update('Escriba un nombre de usuario');
         divMensaje.show();
         return;
      } else if (pass == '') {
         divMensaje.update('Escriba una contraseña');
         divMensaje.show();
         return;
      }

      divMensaje.hide();

      new Ajax.Request('login', {
         method: 'post',
         parameters: {
            'tipo': 'login',
            'nombre': nombre,
            'pass': pass          },
         onSuccess: function(args) {
            var json = (args.responseText || "{}").evalJSON();

            if (json && json.exito) {
               document.location.href = 'visor.html';
            } else {
               divMensaje.update('Error al comunicarse con el servidor');
               divMensaje.show();
               return;
            }
         },
         onFailure: function(args) {
            divMensaje.update('Usuario o contraseña incorrectos');
            divMensaje.show();
         }
      });
   });

   return false;});
```

registro.js

```
document.on('dom:loaded', function(e) {
```

```
    $('registrar').on('click', function(e) {
        var rolSeleccionado = $$('input:checked[name="rol"]').pluck('value');
        if (rolSeleccionado.length) {
            rolSeleccionado = rolSeleccionado[0];
        }
        new Ajax.Request('registro', {
            method: 'post',
            parameters: {
                tipo: 'registro',
                nombre: $('usuario').value,
                pass: $('contrasena').value,
                correo: $('correo').value,
                telefono: $('telefono').value,
                rol: rolSeleccionado || 'USR'
            },
            onSuccess: function(response) {
            }
        });
    });
});
```

index.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <link rel="stylesheet" type="text/css" href="css/estilos.css" />
      <script type="text/javascript" src="js/prototype.js"></script>
      <script type="text/javascript" src="js/login.js"></script>
   </head>
   <body>
      <div id="main_container">
         <div id="header">
            <div class="logo"><img src="img/logo.gif" border="0" alt="" title="" /></div>
         </div>
         <div class="menu">
            <ul>
               <li class="selected"><a href="#">home</a></li>
            </ul>
         </div>

         <div class="center_content">

            <div class="center_left">
               <div class="title_welcome"><span class="red">Ecuavisor</span> Monitoreo
hecho simple</div>
               <div class="features">
                  <div class="title">Características</div>

                  <ul class="list">
                     <li><span class="numero">1</span><span class="texto">Lorem ipsum
dolor sit amet, consectetur adipisicing elit, sed do eiusmod</span></li>
                     <li><span class="numero">2</span><span class="texto">Lorem ipsum
dolor sit amet, consectetur adipisicing elit, sed do eiusmod</span></li>
                     <li><span class="numero">3</span><span class="texto">Lorem ipsum
dolor sit amet, consectetur adipisicing elit, sed do eiusmod</span></li>
                     <li><span class="numero">4</span><span class="texto">Lorem ipsum
dolor sit amet, consectetur adipisicing elit, sed do eiusmod</span></li>
                  </ul>
               </div>
            </div>

            <div class="center_right">

               <div class="text_box">
                  <form id="frm-login" name="frm-login">
                     <div class="title">Iniciar Sesión</div>
```

```
                    <div class="login_form_row">
                        <label class="login_label">Nombre de usuario:</label><input
type="text" name="nombre" id="nombre" class="login_input" />
                    </div>

                    <div class="login_form_row">
                        <label class="login_label">Contraseña:</label><input
type="password" name="pass" id="pass" class="login_input" />
                    </div>
                    <div id="mensaje-error" style="display: none;" class="msj-
error">Usuario o contraseña incorrectos</div>
                    <div class="login_form_row" style="text-align: center">
                        <input type="submit" id="enviar" class="login" value="Enviar datos"
onclick="return false;"/>
                    </div>
                </form>
            </div>

        </div>
        <div class="clear"></div>

    </div>
    <div id="footer">
        <div class="right_footer"><a href="http://csstemplatesmarket.com"
target="_blank"><img src="img/csstemplatesmarket.gif" border="0" alt="" title="" /></a>
        </div>
    </div>
  </div>
 </body>
</html>
```

visor.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <link rel="stylesheet" type="text/css" href="css/estilos.css" />
      <link rel="stylesheet" type="text/css" href="css/camaras.css" />
      <script type="text/javascript" src="js/prototype.js"></script>
      <script type="text/javascript" src="js/camaras.js"></script>
   </head>
   <body>
      <div id="main_container">
         <div id="header">
            <div class="logo"><img src="img/logo.gif" border="0" alt="" title="" /></div>
         </div>
         <div class="menu">
            <ul>
               <li><a href="index.html">home</a></li>
               <li  class="selected"><a href="#">Cámaras</a></li>
            </ul>
         </div>

         <div class="center_content">
            <div class="title_welcome">Mis Cámaras</div>
            <div class="error-usuario" style="display: none;">
               <span>Para poder ver esta página, debe <a href="index.html">iniciar
sesión</a>.</span>
            </div>
            <div id="contenedor-camaras">
            </div>
         </div>


         <div id="footer">
            <div class="right_footer"><a href="http://csstemplatesmarket.com"
target="_blank"><img src="img/csstemplatesmarket.gif" border="0" alt="" title="" /></a>
            </div>
         </div>
      </div>
   </body>
</html>
```

registro.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <link rel="stylesheet" type="text/css" href="css/estilos.css" />
      <script type="text/javascript" src="js/prototype.js"></script>
   </head>
   <body>
      <div id="main_container">
         <div id="header">
            <div class="logo"><img src="img/logo.gif" border="0" alt="" title="" /></div>
         </div>
         <div class="menu">
            <ul>
               <li class="selected"><a href="#">home</a></li>
               <li><a href="#">about</a></li>
               <li><a href="#">demo</a></li>
               <li><a href="#">license</a></li>
               <li><a href="#">modules</a></li>
               <li><a href="#">themes</a></li>
               <li><a href="#">contact</a></li>
            </ul>
         </div>

         <div class="center_content">

            <div class="center_left">
               <div class="title_welcome">Registro de usuarios</div>

               <div class="features formulario">
                  <div class="title">Datos del usuario</div>
                  <p><label for="usuario">Usuario:</label><input type="text" id="usuario"
name="usuario" /></p>
                  <p><label for="contrasena">Contraseña:</label><input type="text"
id="contrasena" name="contrasena" /></p>
                  <p><label for="correo">Correo:</label><input type="text" id="correo"
name="correo" /></p>
                  <p><label for="telefono">Teléfono:</label><input type="text"
id="telefono" name="telefono" /></p>
               </div>

               <div class="features roles">
                  <div class="title">Rol del usuario</div>
                  <label><input type="radio" name="rol" value="ADM"
/>Administrador</label>
```

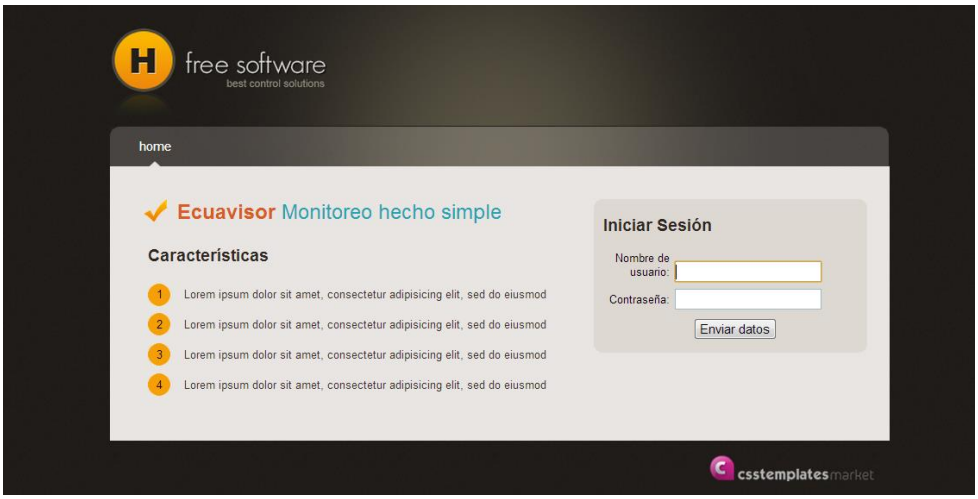
```
                    <label><input type="radio" name="rol" value="USR"
checked="checked"/>Usuario normal</label>
                </div>
                <div class="features">
                    <div class="title">Cámaras</div>
                    <div class="news_box">
                        <div class="news_icon"></div>
                        <div class="news_content">
                            “Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
                        </div>
                    </div>
                    <div class="news_box">
                        <div class="news_icon"></div>
                        <div class="news_content">
                            “Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
                        </div>
                    </div>

                </div>
                <button id="registrar">Registrar</button>
            </div>


            <div class="center_right">

                <div class="software_box"><img src="img/3dbox.gif" alt="" title=""
/></div>

                <div class="text_box">
                    <form id="frm-login" name="frm-login">
                        <div class="title">Iniciar Sesión</div>
                        <div class="login_form_row">
                            <label class="login_label">Nombre de usuario:</label><input
type="text" name="nombre" id="nombre" class="login_input" />
                        </div>

                        <div class="login_form_row">
                            <label class="login_label">Contraseña:</label><input
type="password" name="pass" id="pass" class="login_input" />
                        </div>
                        <div id="mensaje-error" style="display: none;" class="msj-
error">Usuario o contraseña incorrectos</div>
                        <div class="login_form_row" style="text-align: center">
                            <input type="submit" id="enviar" class="login" value="Enviar datos"
onclick="return false;"/>
```

```
                </div>
            </form>
        </div>

        <div class="testimonials">
            <div class="title">Testimonials</div>
            <div class="text_box">
                <p class="testimonial">
                    “Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Ut enim ad minim veniam”<br />
                    <strong>Edward Caloryd</strong>

                </p>


            </div>

        </div>


    </div>

    <div class="clear"></div>

</div>


<div id="footer">
    <div class="left_footer"><a href="#">home</a> <a href="#">about</a> <a href="#">privacy policy</a><a href="#">contact</a></div>
    <div class="right_footer"><a href="http://csstemplatesmarket.com" target="_blank"><img src="img/csstemplatesmarket.gif" border="0" alt="" title="" /></a>
    </div>
</div>
</div>
</body>
</html>
```

Anexo 2

Pantallas

Pantalla de inicio de la aplicación web con dominio www.ecuavisor.com



Pantalla de la visualizacion de las cámaras IP colocadas en un domicilio.