EMPRESA DE TECNOLOGÍAS
DE LA INFORMACIÓN
PARA LA DEFENSA

Aplicación de monitoreo de cámaras en dispositivos móviles para el sistema de Video Vigilancia Digilante.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Cadete Grettel Hevia Cárdenas

Tutores: Ing. Lazaro Jesús Marin Scull.

Ing. Richel Labrada Quiala

Cotutor: Ing. Luis Ramón Samada De la Paz.

Ing. Máxora Rorayma Castro Pérez

La Habana, Julio 2018.

Año 60 de la Revolución.

DECLARACIÓN DE AUTORÍA:

Declaramos ser autor(es) de la presente tesis y reconocemos a la XETID, Empresa de Tecnologías de la Información para la Defensa, los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los __ días del mes _____ del año _____.

Grettel Hevia Cárdenas

_____

Firma de Autor

Ing. Lazaro Jesús Marin Scull

_____

Firma de Tutor

Ing. Richel Labrada Quiala.

_____

Firma de Tutor

Ing. Luis Ramón Samada De la Paz

_____

Firma de Cotutor

Ing. Máxora Rorayma Castro Pérez

_____

Firma de Cotutor

DATOS DEL CONTACTO

Nombre y apellidos del autor: Grettel Hevia Cárdenas

Institución: Empresa de Tecnologías e Información para la Defensa (XETID).

Correo electrónico: ghevia@estudiantes.uci.cu.

Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2 ½, Boyeros, La Habana, Cuba.

Síntesis de Tutores:

Nombre y apellidos de tutor: Lazaro Jesús Marin Scull.

Institución: Empresa de Tecnologías e Información para la Defensa (XETID).

Título: Ingeniero en Ciencias Informáticas.

Correo electrónico: ljmarin@xetid.cu

Graduado en julio de 2013. Desarrollador de Lógica de Negocio de la Línea de Sistemas Embebidos del Centro de Tecnologías SCADA en la Empresa de Tecnologías de la Información para la Defensa (XETID).

Nombre y apellidos de tutor: Richel Labrada Quiala.

Institución: Empresa de Tecnologías e Información para la Defensa (XETID).

Título: Ingeniero en Ciencias Informáticas.

Correo electrónico: rquiala@xetid.cu

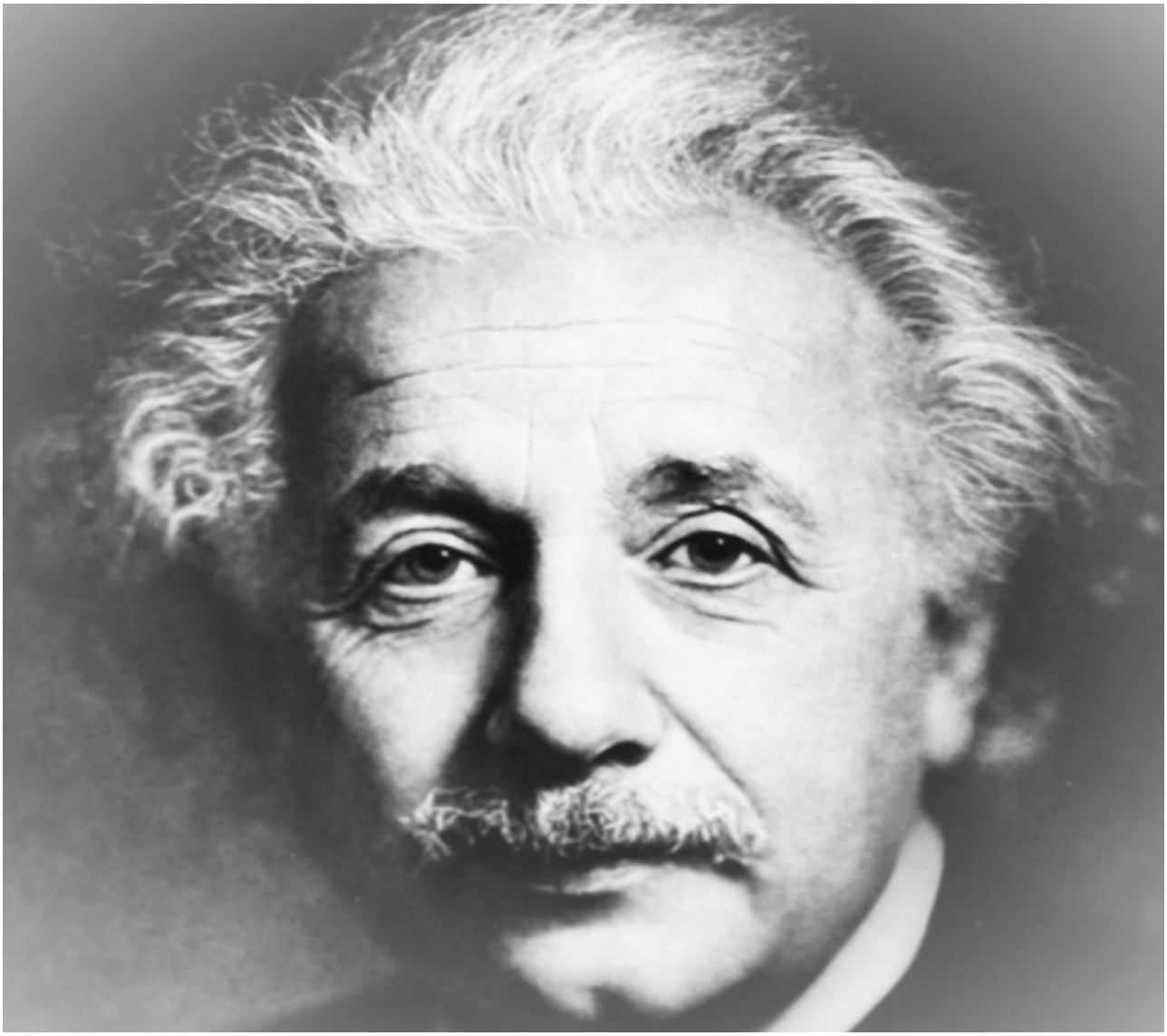
Graduado en junio 2013. Desarrollador, Proyectista y Jefe de Proyectos de Seguridad en el Centro de Servicios Integrales 2015. Desde 2016 Especialista de Producción de la Dirección de Automática y Comunicaciones en la Empresa de Tecnologías de la Información para la Defensa (XETID).

Síntesis de Cotutor:

Nombre y apellidos de tutor: Luis Ramón Samada de la Paz.
Institución: Empresa de Tecnologías e Información para la Defensa XETID.

Título: Ingeniero en Ciencias Informáticas.
Correo electrónico: lrsamada@xetid.cu
Ingeniero en Ciencias Informáticas graduado en la UCI en el año 2011. Tiene 6 años de experiencia en el desarrollo del software y 4 años de experiencia en el tema. Jefe de Línea.

Nombre y apellidos de tutor: Máxora Rorayma Castro Pérez
Institución: Universidad de las Ciencias Informáticas
Título: Ingeniero en Ciencias Informáticas.
Correo electrónico: mcastrop@uci.cu
Ingeniero en Ciencias Informáticas, graduado en la UCI en el año 2007. Profesora del Departamento de Ciencias Básicas de la Facultad I. Graduada de Maestra en Ciencias de la Universidad de La Habana, en la especialidad de Matemática.



"El genio se hace con un 1% de talento y un 99% de trabajo"

Albert Einstein

Agradecimientos:

A mis tutores, nunca serán suficiente los agradecimientos por tanta dedicación y paciencia. Sé que han trabajado conmigo con el corazón y me siento dichosa de que hayan sido ustedes. Mil gracias porque sin su apoyo no hubiese sido posible y por haber estado en el momento más difícil de mi vida.

A mis padres por hacer que me sienta segura aún en los momentos más difíciles, por confiar aun cuando todo parecía perdido y por tanto derroche de paciencia, de amor y de apoyo. Muchas gracias por estar siempre, por guiar mi camino y porque sé que hoy son las personas más felices y orgullosas que comparten junto a mi este triunfo.

A mis abuelos y tíos por velar la llegada de este momento llenos de orgullo y felicidad, y por conspirar junto a mis padres para levantarme de mi reciente caída, seguir adelante y estar hoy en el lugar donde estoy.

A mi novio que sin importar el tiempo ha ocupado un lugar muy importante en mi vida. Gracias por hacerme sonreír cuando más lo necesité, por ayudarme a levantar, por darme ánimo, apoyo y confianza cada momento desde que te conocí.

A mis amigas Laura y Adalid porque ser el mismo resultado de diferentes causas como dijera alguien, nos llevó a construir la linda amistad que hemos compartido. Ha sido un placer cada momento de risa con dolor abdominal, cada momento de ánimo, de confianza, de consuelo, de regaños y de secretos compartidos. Les agradezco por los momentos mágicos y por no darme la espalda en el peor de ellos también.

A Mauro por acompañarme siempre sin importar lo difícil de cada época, por tanta fuerza, luz y espiritualidad en los mejores momentos de estos cinco años y también en los momentos más difíciles. Por estar siempre para mí incondicionalmente, mil gracias.

A Amalita por recordarme siempre cuál es el camino, por traerme de regreso al planeta Tierra incontables veces y por su paciencia y dedicación.

A mi padrino querido por seguirme los pasos siempre de cerca, por su constancia y entrega, por sus ánimos y porque siempre estás ahí cuando te necesito.

Al claustro de profesores de mi facultad por la formación que he recibido como profesional durante los cinco años de mi carrera.

A los profesores que han estado pendiente de mi proceso de recuperación, brindándome su apoyo y la seguridad de poder contar con ellos, para hoy poder estar donde estoy en perfectas condiciones.

Dedicatoria:

A mis padres por anhelar este momento cada minuto desde que inicié mi carrera y por acompañarme paso a paso durante este largo camino transmitiéndome seguridad, apoyo incondicional y todo su corazón.
A Rafael Alaba, Juan José y Nemesia por estar siempre sin reparos ni condición y transmitirme seguridad y armonía.
A Amalita por luchar contra viento y mareas pese a mi sistema imaginativo para poder llegar a donde estoy.

Resumen

A partir de la incorporación de las tecnologías a las organizaciones empresariales se han empleado distintos mecanismos asociados al desarrollo de sistemas de video vigilancia. El presente trabajo de diploma tiene como objetivo desarrollar una aplicación con tecnología Android capaz de visualizar las cámaras del sistema de Video Vigilancia Digilante. La misma permite: gestionar y configurar las cámaras de seguridad, notificar ocurrencia de eventos en tiempo real y el almacenamiento de videos y fotos. La propuesta se realizó siguiendo los pasos del proceso de desarrollo de software que utiliza la Empresa de Tecnologías de la Información para la Defensa, que se basa en la agregación de componentes en cada iteración y está definido en la versión 1.5 del Prodesoft. La solución se desarrolló con el lenguaje de programación Java y con el entorno de desarrollo integrado de programación Android Studio. Se obtuvo el diseño de una arquitectura para la visualización de las cámaras del sistema Digilante, que permite la comunicación con el servidor de media encargado de proveer el flujo de video de las cámaras del sistema de video vigilancia. La aplicación desarrollada es soportada por varias versiones del sistema operativo Android. Se considera que la solución contribuye a elevar el nivel de seguridad en la empresa para la cual fue desarrollada y puede ser desplegada en escenarios con condiciones similares.

Palabras clave: aplicación Android, sistema operativo Android, video vigilancia

Índice

Índice de figuras

Índice de tablas

Introducción

Desde la segunda mitad del siglo veinte las tecnologías se han incorporado paulatinamente a las organizaciones empresariales. Se han empleado como parte de su evolución, diversos mecanismos asociados al desarrollo de sistemas de video vigilancia a fin de contribuir a la convivencia social y prevenir las faltas e infracciones que atenten contra la seguridad pública. Estos avances técnicos irrumpen en el ámbito corporativo permitiendo obtener una serie de ventajas, que no son únicamente de organización y gestión, sino también de vigilancia. Dichos sistemas han pasado de la captura y transmisión de video en forma analógica, hasta el uso de técnicas que permiten capturar video en forma digital y transmitirlo con esquemas distribuidos sobre infraestructuras IP (*Internet Protocol (Protocolo de Internet),* por sus siglas en inglés). [1]

La video vigilancia es en la actualidad uno de los sistemas de seguridad más empleados en el mundo. Utilizada inicialmente en Europa y en América del Norte, se ha expandido hacia los cinco continentes, convirtiéndose en una de las principales herramientas al servicio de la seguridad ciudadana**.** [2] Se considera que tienen un alto nivel de disuasión y prevención, permitiendo condicionar la decisión de un intruso en el momento de irrumpir o no en una determinada instalación.[3] El desarrollo de la video vigilancia se ha extendido en el mercado hasta alcanzar la tecnología móvil, en concreto aquellos *smartphones* que soportan el sistema operativo Android, proporcionando funcionalidades antes impensables en los teléfonos móviles, y soportando, por ejemplo, la transmisión y recepción de vídeo. La necesidad de garantizar la seguridad de personas y bienes en entornos cada vez más poblados, explica la expansión y desarrollo que están experimentando los sistemas de vigilancia en todo el mundo.[4]

En Cuba, en las últimas décadas se ha logrado un notable avance en los sistemas automáticos de vigilancia.[4] La utilización de sistemas de seguridad avanzados ha impulsado a compañías especializadas en el desarrollo de software a elaborar productos de este tipo para el país. Son disímiles las empresas que utilizan los sistemas de video vigilancia permitiendo, sobre todo la calidad de imagen, monitoreo remoto y la seguridad. DATYS, figurando como una de las mayores empresas, incursionó en el desarrollo de un sistema de video vigilancia basada en tecnología IP llamado XYMA SAFE VISION. También en la Universidad de las Ciencias Informáticas se encuentra el Departamento de Desarrollo de Componentes, especializado en el desarrollo de software, donde se implementó el Sistema de Video Vigilancia Xilema Suria. Este sistema constituye una plataforma profesional para la gestión de video vigilancia en cualquier

entorno necesitado de seguridad y protección, y cuenta además con un alto grado de escalabilidad y adaptabilidad. [5] En la Universidad de las Ciencias Informáticas radica a su vez, la Empresa de Tecnologías de la Información para la Defensa (XETID), que presenta dentro de su cartera de productos y servicios un sistema de video vigilancia nombrado Digilante, desarrollado para instituciones de las fuerzas armadas y diversos sectores de la economía. El sistema constituye una herramienta de apoyo al servicio de guardia y permite a los operadores del sistema controlar el área de manera más abarcadora y precisa. Además, se encarga del monitoreo y control de varias cámaras situadas en determinadas zonas de la empresa desde un mismo punto de control.

Aunque Digilante constituye un beneficio para la institución, presenta las siguientes limitantes:

- El oficial de guardia permanece dependiente de la estación de monitoreo durante toda la jornada, lo que impide el recorrido habitual del servicio de guardia o el control de un hecho extraordinario y el monitoreo de las cámaras de manera simultánea.
- El oficial no tiene la posibilidad de visualizar las cámaras desde otro dispositivo que no tenga el mismo sistema operativo donde se ejecuta Digilante: Debian 8 Wheezy.
- La jefatura no puede supervisar el servicio de guardia de otra manera que no sea trasladándose hasta el punto de control, lo cual compromete la seguridad del servicio.

De acuerdo a la situación problemática expuesta anteriormente se identifica como **problema a resolver:** ¿Cómo monitorear las cámaras del Sistema de Video Vigilancia Digilante garantizando su portabilidad[1]?

El objeto de estudio se centra en **objeto de estudio**: los Sistemas de Video Vigilancia en dispositivos móviles, enmarcado en el **campo de acción:** Sistemas de Video Vigilancia en dispositivos móviles con sistema operativo Android.

Objetivo general: Desarrollar una aplicación informática para dispositivos móviles con sistema operativo Android que permita el monitoreo de cámaras del sistema de Video Vigilancia Digilante garantizando su portabilidad.

[1] Posibilidad que tiene el vigilante de trasladarse de un lugar a otro portando el sistema en un dispositivo más manuable.

Objetivos específicos:

1. Fundamentar teóricamente el estado del arte y elementos propios de las herramientas de monitoreo de cámaras de video vigilancia.
2. Diseñar una aplicación informática para dispositivos móviles que permita el monitoreo de cámaras del sistema de Video Vigilancia Digilante garantizando su portabilidad.
3. Implementar una aplicación informática para dispositivos móviles que permita el monitoreo de cámaras del sistema de Video Vigilancia Digilante garantizando su portabilidad.
4. Evaluar la aplicación informática para dispositivos móviles que permita el monitoreo de cámaras del sistema de Video Vigilancia Digilante garantizando su portabilidad.

Idea a defender:

Si se desarrolla una aplicación informática para dispositivos móviles con sistema operativo Android que permita el monitoreo de cámaras del sistema de Video Vigilancia Digilante garantizando su portabilidad entonces se contribuye a elevar la eficiencia de la seguridad en la XETID.

Para dar cumplimiento a los objetivos específicos se propone ejecutar las siguientes **tareas de investigación**:

1 Revisión bibliográfica para la identificación y análisis de los conceptos asociados al objeto de estudio.

2 Análisis de la arquitectura del Sistema de Video Vigilancia Digilante.

3 Análisis de herramientas Android para el monitoreo de cámaras de video vigilancia.

4 Selección de las herramientas, tecnologías y proceso de desarrollo de software para implementar la herramienta.

5 Definición de los requisitos funcionales y no funcionales del sistema.

6 Implementación de la *APK*[2] para el monitoreo de cámaras de video vigilancia.

[2] (**A**ndroid A**p**plication Pac**k**age, por sus siglas en inglés) es un archivo mediante el cual se pueden instalar aplicaciones Android.

7 Diseño y realización de las pruebas correspondientes a los requerimientos implementados, para detectar no conformidades.

8 Evaluación del sistema con vistas a lograr un buen funcionamiento.

Para el desarrollo del presente trabajo se emplean los siguientes métodos científicos:

Métodos teóricos

- **Analítico Sintético**: se realiza un estudio de todos los aspectos relacionados a los sistemas de video vigilancia, facilitando la comprensión del tema a desarrollar y obteniendo una síntesis del análisis realizado y definir las características generales del sistema.
- **Histórico lógico**: permite estudiar de forma analítica la evolución y desarrollo de los sistemas de video vigilancia sobre dispositivos móviles, facilitando la identificación de las características que los distinguen.
- **Modelación:** se generaliza en la elaboración de los diagramas de clases y del negocio, diagramas de despliegue, diagramas de actividades y diagramas de componentes. Garantiza una vista general del sistema, así como una representación de las características del problema.

Método empírico:

- **Entrevista:** se realizan entrevistas periódicas a los líderes y desarrolladores del sistema de Video Vigilancia Digilante para la recopilación de datos e información que ayude a identificar el problema a resolver. Ver **Anexo 1**Tabla 1. Entrevista realizada al Ing. Luis Ramón Samada de la Paz, especialista en temas de video vigilancia del centro de Tecnologías de SCADA y Seguridad.

Para la presentación de los resultados obtenidos en el presente trabajo, el informe consta de 4 capítulos estructurados de la siguiente forma:

Capítulo 1. “Fundamentación teórica sobre los sistemas de vídeo vigilancia para dispositivos móviles”: el capítulo contiene los conceptos fundamentales asociados al tema, así como un estudio del estado del arte de aplicaciones móviles con sistema operativo Android que permitan el monitoreo de cámaras de video vigilancia; con el fin de realizar una comparación con la solución a desarrollar. Se hace referencia a las diferentes tecnologías, lenguajes y herramientas a emplear para el desarrollo de la aplicación.

Capítulo 2. “Propuesta de la aplicación informática para el monitoreo de cámaras del sistema de Video Vigilancia Digilante”: en este capítulo se realiza el análisis del modelo de dominio del problema y una caracterización de la solución propuesta. Se especifican los requisitos funcionales y no funcionales definidos según las necesidades de la empresa y el modelo conceptual elaborado. Se definen los conceptos fundamentales del trabajo investigativo y se diseña la arquitectura de la solución.

Capítulo 3. “Diseño de la aplicación informática para el monitoreo de cámaras del sistema de Video Vigilancia Digilante”: en este capítulo Se describe la arquitectura de los módulos a partir del diagrama de clases de diseño. Se elabora el modelo de la base de datos, mediante el diagrama de entidad-relación. Además, se definen los patrones de diseño, como base para la búsqueda de soluciones en el proceso de desarrollo de software.

Capítulo 4. “Implementación y evaluación de la aplicación informática para el monitoreo de cámaras del sistema de Video Vigilancia Digilante”: En este capítulo se elabora el diagrama de componente y el de despliegue, describiendo en cada uno los elementos que lo conforman. Incluye la implementación de la aplicación según los requisitos propuestos. Se realizan las pruebas a la aplicación implementada para asegurar el correcto funcionamiento del software.

Capítulo 1: Fundamentación teórica sobre los sistemas de video vigilancia para dispositivos móviles

En el presente capítulo se exponen las principales características de los sistemas de video vigilancia en dispositivos móviles y conceptos fundamentales relacionados con el tema de la investigación. Se realiza un estudio sobre los sistemas homólogos en el mercado; y se definen las tecnologías y herramientas que son utilizadas en la modelación, el diseño e implementación de la solución.

1.1 Conceptos relacionados

Video vigilancia: la video vigilancia es un método de supervisión por imágenes para detectar y registrar hechos extraordinarios en tiempo real que pueden ser utilizadas en casos legales o de auditoría.[6]. Según Francisco Klauser "la video vigilancia es 'conservadora' y 'protectora'; conservadora porque se encarga de la conservación del orden público y de la prevención de la conducta antisocial, y protectora pues protege diferentes áreas del peligro" [7]. [8]

Sistema de video vigilancia: los sistemas de video vigilancia permiten la gestión de múltiples cámaras de seguridad para el control y supervisión de instalaciones locales y remotas. Se trata básicamente de un grabador digital con capacidad de almacenamiento de imágenes y sonido ininterrumpido durante las 24 horas, que muestra los resultados en un televisor o monitor. Un buen sistema debe proporcionar imágenes de calidad tanto diurnas como nocturnas, ser flexible y fácil de usar y proporcionar imágenes para grabar evidencias o para ayudar al análisis de cualquier incidente. [9]

Cámara IP: el uso de cámaras IP para video vigilancia se conoce como video vigilancia IP, siendo una efectiva solución de seguridad que ofrece monitorización y control avanzado en sistemas de seguridad. Son dispositivos autónomos que cuentan con un servidor de video incorporado que les permite transmitir las imágenes capturadas a través de redes IP. Las cámaras IP permiten al usuario tener la cámara en una localización y ver el video digital, en tiempo real, desde otro lugar. [10]

***Streaming*:** capacidad de distribuir contenidos multimedia a través de una red digital, con la característica especial de permitir el acceso a estos contenidos según se requiera, sin necesidad de descargarlos previamente. Las aplicaciones basadas en *streaming* pueden clasificarse en aquellas relacionadas con la interacción entre dos o más usuarios (como video conferencia y transmisión de voz). [11]

Servicio Web: se define como un sistema de software diseñado para soportar la interoperabilidad máquina - máquina a través de una red. Los servicios que se ofrecen a través de la Web, se caracterizan por ofrecer

una interfaz especialmente ágil y flexible, como pueden ser todos los servicios ofrecidos entre otros por las grandes empresas de Internet. [12]

REST: *REST* es el acrónimo de Transferencia de Estado Representacional, término usado por Roy Fielding (uno de los creadores de *HTTP*[3]) para describir un estilo de arquitectura que utiliza como modelo en los sistemas de computación Web. No es un estándar, sino un enfoque que muestra cómo desarrollar y proporcionar servicios en Internet, por tanto, considerado como un estilo arquitectónico para diseño de software a gran escala. [13]

SOAP: *SOAP*, Protocolo de Acceso de Objeto Simple (*Simple Object Access Protocol*, por sus siglas en inglés). Es el protocolo de mensajería estándar utilizado por los servicios web. Su aplicación principal es la comunicación entre aplicaciones. Codifica el uso de XML como un esquema de codificación para los parámetros de solicitud y respuesta utilizando HTTP como medio de transporte. [14]

JSON: *JavaScript Object Notation* (notación de objetos JavaScript), es un nuevo formato de datos construido sobre JavaScript, propuesto por el experimentado ingeniero de software Douglas Crockford. Es un formato de datos muy ligero basado en un subconjunto de la sintaxis de JavaScript: literales de matrices y objetos. Como usa la sintaxis JavaScript, las definiciones JSON pueden incluirse dentro de archivos JavaScript y acceder a ellas sin ningún análisis adicional como los necesarios con lenguajes basados en XML. Su sintaxis no es nada más que la mezcla de literales de objeto y matrices para almacenar datos. JSON representa solamente datos, no incluye el concepto de variables, asignaciones o igualdades. [15]

Android: Android es un sistema operativo y una plataforma de software, basado en Linux para teléfonos móviles. Es utilizado por *tablets*, *netbooks*, reproductores de música e incluso máquinas de escritorio. Permite programar en un entorno de trabajo (*framework*) de Java, aplicaciones sobre una máquina virtual *Dalvik* (una variación de la máquina de Java con compilación en tiempo de ejecución). Lo que lo diferencia de otros sistemas operativos, es que cualquier persona que sepa programar puede crear nuevas aplicaciones, *widgets*[4], o incluso, modificar el propio sistema operativo, dado que Android es de código libre,

---

[3] Protocolo de Transferencia de Hipertexto (*Hypertext Transport Protocol*, por sus siglas en inglés)

[4] Un *widget* es una pequeña aplicación que facilita el acceso a funciones frecuentes.

por lo que sabiendo programar en lenguaje Java, va a ser muy fácil comenzar a programar en esta plataforma. [16]

***Smartphone*:** como su nombre lo indica, teléfono inteligente, es un dispositivo que ofrece más funciones que un teléfono celular común o sea que posee características similares a las de un computador. Utilizan cualquier interfaz para el ingreso de datos, como por ejemplo el teclado presente en la pantalla táctil o el QWERTY que no es más que el teclado convencional. Permiten leer documentos en distintos formatos, entre ellos los PDFs y archivos de Microsoft Office. Poseen una cámara digital y constan de un sistema de conexión inalámbrica, ya sea Bluetooth o WiFi y un Sistema de Posicionamiento Global (*GPS)*. [15]

Los *smartphones* utilizan un sistema operativo que puede ser tanto *Symbian OS*, como *BlackBerry OS*, Android*, iOS*, o *Windows Mobile*. Permiten la instalación de programas adicionales para el procesamiento de datos y la conectividad, contando con un procesador con la capacidad y la potencia para manejar tareas avanzadas.[17]

Red *WIFI*: *Wi-Fi* son las tecnologías de comunicación inalámbrica mediante ondas, también llamada *WLAN* (*Wireless LAN*, Red inalámbrica) o estándar IEEE 802.11. *WI-FI* no es una abreviatura de *Wireless Fidelity*, simplemente es un nombre comercial. [18]

Las redes *Wi-Fi* poseen una serie de ventajas, entre las cuales podemos destacar:

Al ser redes inalámbricas, la comodidad que ofrecen es muy superior a las redes cableadas porque cualquiera que tenga acceso a la red puede conectarse desde distintos puntos dentro de un rango suficientemente amplio de espacio. [18]

Una vez configuradas, las redes *Wi-Fi* permiten el acceso de múltiples ordenadores sin ningún problema ni gasto en infraestructura, no así en la tecnología por cable. [18]

La *Wi-Fi Alliance* asegura que la compatibilidad entre dispositivos con la marca *Wi-Fi* es total, con lo que en cualquier parte del mundo podremos utilizar la tecnología *Wi-Fi* con una compatibilidad total. [18]

Pero como red inalámbrica, la tecnología *Wi-Fi* presenta los problemas intrínsecos de cualquier tecnología inalámbrica. Algunos de ellos son:

El sistema *Wi-Fi*, posee una menor velocidad en comparación a una conexión por cables, debido a las interferencias y pérdidas de señal que el ambiente puede acarrear. [18]

La desventaja fundamental de estas redes existe en el campo de la seguridad. Existen algunos programas capaces de capturar paquetes, trabajando con su tarjeta *Wi-Fi* en modo promiscuo, de forma que puedan calcular la contraseña de la red, y de esta forma acceder a ella. Las claves de tipo *WEP* son relativamente fáciles de conseguir con este sistema. [18]

La alianza *Wi-Fi* arregló estos problemas sacando el estándar *WPA* y posteriormente *WPA2*, basados en el grupo de trabajo 802.11i. Las redes protegidas con *WPA2* se consideran robustas dado que proporcionan muy buena seguridad. De todos modos, muchas compañías no permiten a sus empleados tener una red inalámbrica. [18]

Protocolos de flujo de video: los protocolos de flujo de video que se emplearon durante el desarrollo de la solución fueron *HTTP*. [19]

***HTTP*:** Protocolo de Transferencia de Hipertexto (*Hypertext Transport Protocol*, por sus siglas en inglés) es el protocolo usado en cada transacción de la web. Mediante él, se envían las peticiones de acceso a una página, la respuesta con el contenido y la información en ambos sentidos. Específicamente en la transmisión de video, este protocolo es usado para que un servidor web pueda enviar los datos a un medio visualizador, permitiéndole al cliente la descarga automática. Algunos modelos de cámaras IP incorporan este protocolo para transmitir los flujos de video capturados. [19]

***UDP*:** el protocolo *UDP* (*User Datagram Protocol,* protocolo de datagrama de usuario) proporciona una comunicación sencilla entre las aplicaciones de dos ordenadores. Al igual que el protocolo *IP*, *UDP* es:

No orientado a conexión. No se establece una conexión previa con el otro extremo para transmitir un mensaje *UDP*. Los mensajes se envían y estos pueden duplicarse o llegar desordenados al destino.

No fiable. Los mensajes *UDP* se pueden perder o llegar dañados. *UDP* utiliza el protocolo *IP* para transportar sus mensajes. No añade ninguna mejora en la calidad de la trasferencia; aunque sí incorpora los puertos origen y destino en su formato de mensaje. Las aplicaciones (y no el protocolo *UDP*) deberán programarse teniendo en cuenta que la información puede no llegar de forma correcta.[20]

***WS*:** el protocolo *WS* (*Websocket*, por sus siglas en inglés) comenzó siendo parte del estándar *HTML*5, pero luego continuó su evolución como un estándar separado. Estas tecnologías en conjunto plantean un nuevo paradigma de diseño y programación de aplicaciones web. Permite mantener una conexión *full-duplex* y con estado entre cliente y servidor web. Aunque pensado para su utilización con JavaScript dentro de un navegador web puede ser extendido para soportar subprotocolos binarios y usarlo desde aplicaciones que se ejecuten fuera de un cliente web.[21]

1.2 Soluciones existentes

Subsistema Visor APK Xilema Suria

El subsistema Visor APK permite el control y visualización de cámaras para el sistema XILEMA Suria 3.0 desde un dispositivo con Sistema Operativo Android. Es capaz de notificar la desconexión a los usuarios autenticados en caso de que el Visor se desconecte del servidor. De igual modo, muestra las cámaras y zonas a las cuales tiene acceso el usuario y visualiza el flujo de video de una cámara. Permite al usuario adicionar una vista personalizada y la cámara que se reproducirá por cada área de visualización. Puede acoplar y desacoplar un área de visualización perteneciente a una vista. Muestra los sensores disponibles para cada flujo de video en dependencia de la cámara. Admite realizar grabaciones manuales, manipular en el visor los parámetros de las cámaras instaladas en el sistema y generar alarmas visuales por eventos asociados en el visor. El usuario puede silenciar una alarma, limpiar alarmas en el servidor y mostrar cambios de estado de la cámara en el visor. Permite hacer una instantánea del video que se está reproduciendo en un área de visualización, guardando la instantánea por defecto en el visor. Además, incorpora un marcador al video que se está reproduciendo en un área de visualización, en tiempo real y permite mostrar un listado de las vistas existentes en el visor. [22]

IP Cam Viewer (Free)

Es una aplicación que se utiliza para conectar cámaras de video vigilancia profesionales y webcams más completas en el mundo de los smartphones. A pesar de tener un diseño sencillo y ser una herramienta muy básica, cuenta con numerosas opciones para dar servicio al usuario respondiendo a cualquier necesidad. Presenta un buen listado de marcas de cámaras de vigilancia para agregar los datos de conexión como la dirección IP y crear el vínculo. Con ello se permite hacer un visionado en tiempo real de lo que capta la cámara, sin importar la distancia a la que se encuentre. Si cuenta con movimientos, el usuario puede redirigir el encuadre a distancia, incluso hacer zoom o activar y desactivar el sensor de movimiento.

Otro aspecto de esta aplicación es la posibilidad de conectar el dispositivo con webcams usuales, siempre y cuando estas estén conectadas a Internet mediante un programa. Permite realizar el visionado de cámaras de vigilancia públicas. Todo ello desde el dispositivo, sólo contando con una buena conexión a Internet. [23]

Entre sus características más destacadas figuran los más de 1600 dispositivos compatibles que posee, incluidos *NVR* y *DVR*. Tiene un amplio protocolo compatible con *RTSP/ONVIF/MMSH* y soporte para *codecs* MPEG4/H264/MJPEG. Ofrece control PTZ (*Pan/Tilt/Zoom*), relé y otras funciones del dispositivo. También cuenta con agrupación de cámaras, secuencia automática y acciones masivas. Exporta e importa usando *sdcard*, email o Dropbox. Brinda protección de contraseñas para aplicaciones, *SSL/HTTPS* y privacidad sin nube. Tiene la capacidad de escanear la cámara para la selección automática del conductor, inicio y bloqueo de *widgets* de pantalla y soporte multiplataforma para todos sus dispositivos móviles.[24]

TinyCam Monitor (Free)

Es una aplicación Android para la vigilancia o control remoto de sus cámaras privadas o públicas de *red/IP*, codificadores de video, *DVR* y *webcams*. Tiene dos versiones: *FREE* y *PRO*, y soporta miles de cámaras *IP MJPEG/MPEG4/H264/H265/RTSP/ONVIF/P2P*. Es compatible con todos los principales proveedores de cámaras *IP*, *DVRs/NVRs* y más de 10.000 cámaras compatibles con *ONVIF Profile S*, incluidas las cámaras chinas baratas. La cámara Android interna (delantera y trasera) también es compatible con el uso de la aplicación como cámara IP o cámara *dashcam*. Funciona con la mayoría de las cámaras *IP* modernas y tiene una grabación continua 24/7. Ofrece soporte de múltiples plataformas: Android, *iOS* (próximamente) y web. Cuenta con 7 días de almacenamiento, descarga de eventos, cronología, programador, video detección de movimiento con máscara y sensibilidad, y detección de nivel sonoro. [25]

TinyCam Monitor es la única aplicación en el mercado que soporta tanto detección de movimiento en la aplicación como en la cámara. Posee detección de rostro, escáner de *LAN* para la detección automática de cámaras, procesamiento en tiempo real de audio (silencio y alarmas) para utilizarlo como monitor de bebé con gráfica de audio, monitorización audio de múltiples cámaras a la vez y múltiples audios de fondo. Brinda soporte de sensores para algunas cámaras; accesos directos para *widgets*, ventanas flotantes y pantalla de inicio; *plugin* de automatización *Tasker/Locale*; soporte para Android *Wear* e interfaz TV.[25]

Para poder establecer una comparación entre las aplicaciones antes descritas, se seleccionaron una serie de aspectos, teniendo en cuenta las características y funcionalidades necesarias que debe cumplir la solución que se desea obtener, como se muestra en la Tabla 1.

Tabla 1. Comparación de las soluciones existentes

Criterios de Comparación	**Subsistema Xilema Suria Visor APK**	**IP Camera Viewer (Free)**	**TinyCam Monitor (Free)**
Código abierto	No	Si	No
Sistema de autenticación	Sí	No	No
Portabilidad[5]	Si	Si	Si
Comunicación con el servidor	XML	Ninguna	Ninguna

Luego de realizado el análisis de las soluciones existentes en cuanto a sus características y funcionalidades; y de efectuar la comparación entre estas de acuerdo a los aspectos requeridos, ver Tabla 2, se concluye lo siguiente:

El análisis del criterio "sistema de autenticación" en las aplicaciones en cuestión arroja que dos de ellas (*IP Camera Viewer* y *TinyCam Monitor*) no poseen esta prestación por lo tanto no brindan seguridad al sistema, mientras que Xilema Suria Visor *APK* además de contar con este servicio ofrece una vista de la interfaz de autenticación brindando experiencia y sirviendo de apoyo al levantamiento de requisitos. Aun así, la comunicación que establece con el servidor para la ejecución de este servicio es a través de XML lo cual resulta trasparente a la solución teniendo en cuenta que la misma debe comunicarse con el servidor mediante *JSON*. Las dos aplicaciones restantes no presentan esta discrepancia puesto que reciben el flujo de video directamente desde las cámaras. El caso específico de IP Camera Viewer a partir del criterio de

[5] Posibilidad que tiene el vigilante de trasladarse de un lugar a otro portando el sistema en un dispositivo más manuable.

"código abierto" ofrece una vista al código permitiendo un análisis del mismo para ganar en estilo de trabajo en aplicaciones Android y en cuanto a organización durante la postrera implementación de la solución. Se determina por ende que, a pesar de ser aplicaciones portables que realizan el monitoreo y control de cámaras de video vigilancia desde dispositivos móviles con sistema operativo Android, no satisfacen las exigencias de la solución propuesta, aunque sí facilitan el trabajo posterior proporcionando elementos que pueden ser reutilizados tales como el estilo de programación y el diseño de las interfaces.

1.3 Proceso de Desarrollo de Software, Prodesoft (v1.5)

Prodesoft constituye una guía para orientar el proceso de desarrollo de aplicaciones informáticas en la XETID. Incluye los procesos propios de la construcción del producto de software y la base para su gestión. Definiendo para ello el ciclo de vida de un proyecto en 5 fases: inicio, modelación, construcción, explotación experimental y despliegue. Cada fase terminará en un hito con el objetivo fundamental de evaluar y decidir el paso a la siguiente fase de desarrollo. [26]

A continuación, se describen las fases de Prodesoft:

- **Inicio:** en la etapa de inicio se definen los objetivos y alcance del proyecto, así como la identificación de involucrados y ejecutores. Se estiman de manera general las actividades a realizar durante el ciclo de desarrollo del mismo (Cronograma General) y se establece la estrategia a seguir para realizar la modelación del negocio y la captura de requisitos. [26]
- **Modelación:** se encarga de la captura de las partes esenciales del sistema, donde se identifican los procesos de negocio fundamentales y se aceptan los requerimientos funcionales, obteniéndose la línea base de la arquitectura y una estrategia de construcción de la aplicación aprobada por los implicados en el proyecto. El hito fundamental es la liberación de la arquitectura de sistema, datos y despliegue. [26]
- **Construcción:** Se aclaran los requisitos restantes y se completa el desarrollo del sistema sobre una base estable de la arquitectura. En esta etapa todas las características, componentes, y requerimientos deben ser integrados, implementados, y probados en su totalidad, obteniendo una versión liberada del producto. [26]

- **Explotación experimental:** Se convierte la versión liberada del producto en una solución estable, donde se eliminan los errores que surgen durante las pruebas y se obtiene una certificación funcional y de seguridad del producto. [26]
- **Despliegue:** Se instala y configura el sistema para un ambiente de producción real, se capacita al personal que usará la aplicación y se continúa dando soporte durante la explotación del sistema, culminando de ser preciso con transferencias tecnológicas. [26]

Las fases de proyecto se descomponen en iteraciones que constituyen un recorrido medianamente completo de todas las disciplinas de desarrollo. Desde el punto de vista ingenieril cada fase desarrolla y evoluciona una parte del producto con un esfuerzo variable en recursos humanos y logísticos. [26]

Con la implementación del Marco Metodológico, Tecnológico y Organizativo para la integración de los procesos de gestión y desarrollo de software (PRODESOFT 1.5) se garantiza: [26]

- La clasificación de los proyectos y las formas de integración.
- La definición del ciclo de vida del proyecto.
- El marco de referencia para la organización de la producción.
- El establecimiento como filosofía de trabajo de las fábricas de software dividiendo la ingeniería de dominio de la ingeniería de aplicativos.
- La definición el enfoque del desarrollo a partir de un modelo híbrido: basado en componentes, iterativo e incremental y con técnicas de prototipado funcional.
- El establecimiento de las disciplinas de desarrollo, roles y artefactos.
- La guía para la ejecución de los procesos de gestión de proyectos.
- La especificación de los procesos y actividades asociadas al soporte.
- Organización y estandarización a nivel institucional en el desarrollo de proyectos de software.
- Software más robustos, predecibles, reutilizables y de fácil mantenimiento.
- Especialización de los equipos en roles y dominios de solución.
- Reducción de tiempo, esfuerzo y costo en la producción de software.

1.4 Ambiente de Desarrollo

Lenguaje de modelado

El lenguaje de modelado que se emplea en el desarrollo del proyecto es UML 2.1 (*Unified Model Language*, por sus siglas en inglés), lenguaje sugerido por Prodesoft por presentar ventajas en el desarrollo del software. Es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de *software.* Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se emplea para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. [27]

Herramienta de modelado

El Visual Paradigm para *UML* es una herramienta Case multiplataforma (*Windows, GNU/Linux, Mac OS X*) que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue además de la última notación *UML* 2.1, ingeniería inversa, generación de código, importación desde *Rational Rose*, exportación/importación *XMI*, generador de informes, editor de figuras, integración con MS Visio, integración *IDE* con *Visual Studio*, *IntelliJ IDEA*, Eclipse*, NetBeans* y otros. [28]. Es una herramienta colaborativa por lo que es posible que varios usuarios trabajen sobre el mismo proyecto y su intuitivo diseño la convierte en una herramienta muy fácil de utilizar.

Se seleccionó esta herramienta para la implementación del proyecto porque permite el desarrollo de diagramas más atractivos y el modelado del entorno del cliente, así como la simulación de sus procesos, y una interfaz que facilita la interacción entre diferentes diagramas. Al ser una herramienta multiplataforma, es utilizable en los sistemas operativos Linux.

Lenguaje de programación

El lenguaje de programación utilizado en el desarrollo de la solución es Java 7. Es una plataforma sencilla y eficaz, destinada al desarrollo y despliegue de aplicaciones securizadas, en sistemas heterogéneos en un entorno distribuido, con un consumo de recursos mínimo y que funciona en cualquier plataforma física y de *software*. Posee una sintaxis simple, orientada a objetos e interpretada, que permite optimizar el tiempo y el ciclo de desarrollo (compilación y ejecución). Las aplicaciones son portables sin modificación en numerosas plataformas físicas y sistemas operativos. Son resistentes porque el motor de ejecución de Java se encarga

de la gestión de la memoria (*Java Runtime Environment*) y resulta más fácil escribir programas sin error debido a un mecanismo de gestión de errores más evolucionado y estricto. En particular las aplicaciones gráficas son eficaces debido a la puesta en marcha y a la asunción del funcionamiento de varios procesos ligeros (*Thread y multithreading*). [29]

IDE

Durante la implementación se utiliza el *IDE* (*Integrated Development Environment*, por sus siglas en inglés) desarrollo *Android Studio* v1.5, basado en *IntelliJ IDEA*. De igual modo se emplea el potente editor de códigos y las herramientas para desarrolladores de *IntelliJ*, *Android Studio* ofrece otras funciones que aumentan la productividad durante la compilación de aplicaciones para Android: [30]

- Sistema de compilación flexible basado en *Gradle*7.
- Un emulador rápido con varias funciones.
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android.
- *Instant Run*, para aplicar cambios mientras la aplicación se ejecuta sin la necesidad de compilar un nuevo *APK*.
- Integración de plantillas de código y *GitHub*, para ayudarte a compilar funciones comunes de las aplicaciones e importar ejemplos de código.

Sistema Gestor de Base de Datos

Se selecciona *SQLite* en su versión 3.8 debido a que es una de las mejores alternativas para la persistencia de los datos sobre texto plano o ficheros en *XML* ya que es un sistema gestor muy ligero que posibilita la portabilidad local de los datos y no requiere de un gran procesamiento en memoria para su gestión. Android posee herramientas muy útiles que facilitan las tareas de manipulación y consulta sobre este tipo de base de datos. [30]

SQLite es un sistema de gestión de bases de datos que enlaza con el programa pasando a ser parte integral del mismo. El programa emplea la funcionalidad de *SQLite* a través de llamadas simples a subrutinas y funciones reduciendo la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más rápidas y seguras que la comunicación entre procesos.[30]

***ORMLite* V4.41**

ORMLite – Paquete (*Lightweight Object Relational Mapping*, por sus siglas en inglés) proporciona algunas funcionalidades simples y livianas para objetos persistentes de Java en bases de datos *SQL*, al tiempo que se evita la complejidad y la sobrecarga de paquetes de *ORM* más estándar. Es fácil de utilizar y ofrece las siguientes características: [31]

- Configuración de sus clases por simple adición de anotaciones Java.
- Potente base de datos abstractos acceso a objetos (DAO) clases.
- Generador de consultas flexibles para construir fácilmente consultas simples y complejas. Soporta *MySQL*, *PostgreSQL, Microsoft SQL Server*, *H2*, *Derby*, *HSQLDB*, y *SQLite* y se puede extender a otras bases de datos con relativa facilidad.
- Brinda apoyo provisional para *DB2*, *Oracle*, *ODBC*, y *Netezza*.
- Maneja "compilado" Sentencias de *SQL* para las tareas de consulta repetitiva.
- Es compatible con "extranjeros" los objetos con el campo de la clase es el objeto, sino un identificador almacenado en la tabla de base de datos. Soporte básico para las transacciones de base de datos.
- Auto genera *SQL* para crear y eliminar tablas de bases de datos.
- Soporte para la configuración de tablas y campos sin anotaciones.
- Admite llamadas nativas de Android *API* de base de datos *SQLite*.

Control de versiones

Subversion es una herramienta de código abierto, multiplataforma (*Win32, Linux, Mac*, etc.), para el control de versiones de ficheros electrónicos, como son el software y la documentación. Se basa en un repositorio central que actúa como un servidor de ficheros con la capacidad de recordar todos los cambios que se hacen tanto en sus directorios como en sus ficheros. El repositorio incrementa un número global de revisión con cada conjunto de cambios enviados (*commit*) al mismo. Es posible copiar y renombrar ficheros; crear una rama del proyecto es tan fácil como copiar un directorio. Se puede pedir una salida con las diferencias entre dos revisiones arbitrarias o que recupere algún sub-árbol de la revisión N.[32]

1.5 Conclusiones del capítulo

Después de realizar el marco teórico del presente capítulo se concluye que:

- El análisis de los conceptos y elementos referentes a los sistemas de video vigilancia en dispositivos móviles, permitió comprender mejor el objeto de estudio de la investigación.
- El estudio de las soluciones existentes arrojó como resultado que las aplicaciones estudiadas sirvieron de apoyo al levantamiento de requisitos.
- El estudio realizado sobre el PRODESOFT y las diferentes tecnologías permitió la creación del entorno de trabajo para el posterior desarrollo de la solución.

Capítulo 2: Propuesta de la aplicación informática para el monitoreo de cámaras del sistema de Video Vigilancia Digilante.

El presente capítulo muestra la descripción de una aplicación para dispositivos móviles con sistema operativo Android. En él se detallan las funcionalidades que serán implementadas y se identifican los requerimientos funcionales y no funcionales del sistema a implementar. Se elaboran los prototipos de interfaz de usuario y se realiza el diseño de la arquitectura, mediante el uso de los estilos y patrones arquitectónicos.

2.1 Modelo de dominio

El modelo de dominio se emplea para representar el vocabulario y los conceptos clave del dominio del problema e identifica las relaciones entre todas las entidades comprendidas en este ámbito. A continuación, se ofrece la tabla y el diagrama que muestra los elementos que componen el modelo de dominio de la propuesta de solución.

Tabla 2. Modelo de dominio

Conceptos	Descripción
Usuario	Es la persona que accede a la aplicación.
PC[6]	Máquina de escritorio donde se encuentra instalado el sistema Digilante.
Digilante	Sistema de Video Vigilancia.

[6] *PC* (*Personal Computer*, por sus siglas en inglés), computadora personal.

Cámaras	Dispositivos autónomos que cuentan con un servidor de video incorporado que les permite transmitir las imágenes capturadas a través de redes.

Diagrama de modelo de dominio.

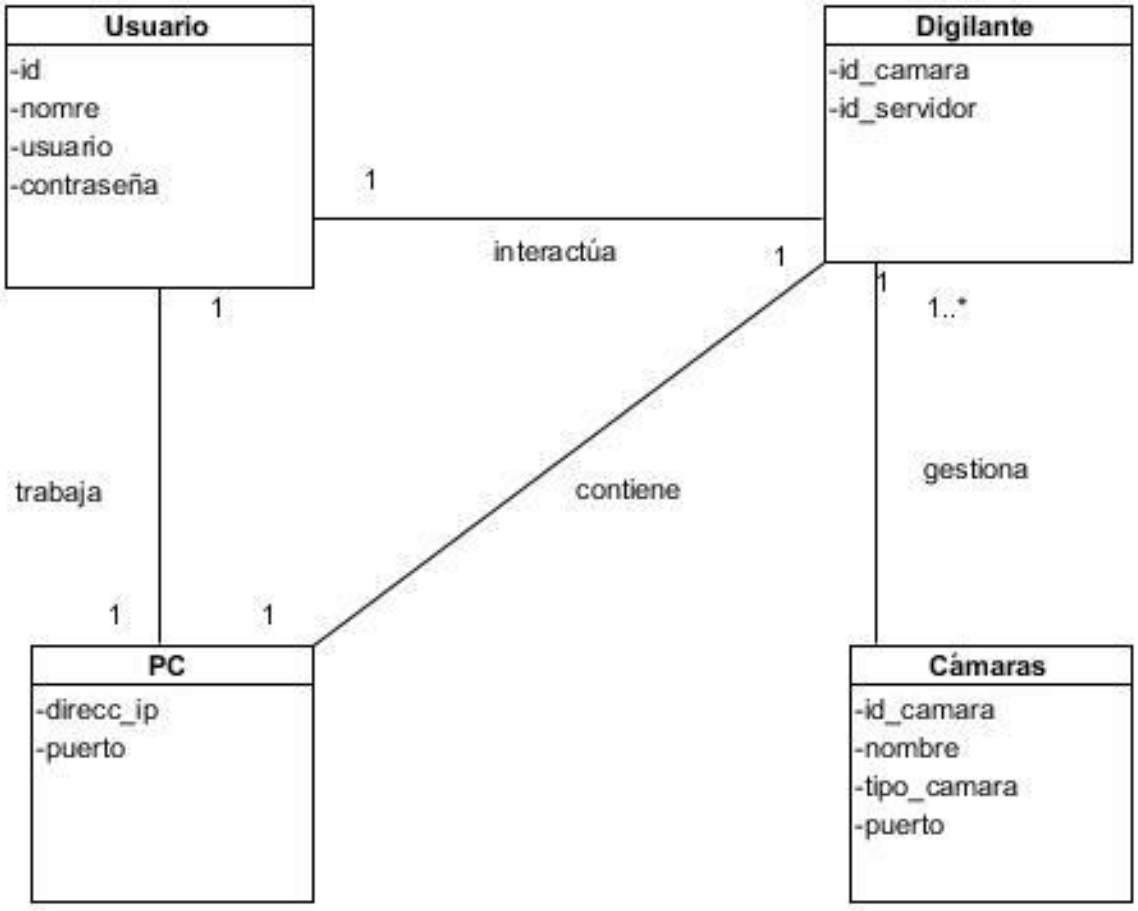


Figura 1. Modelo de Dominio

El modelo obtenido representa el contexto de negocio que se va a informatizar. Aquí, el usuario trabaja directamente con la PC que a su vez contiene al sistema Digilante, por lo cual existe una interacción del usuario con el sistema. Digilante se encarga de la gestión de las cámaras.

2.2 Solución propuesta

En el capítulo anterior se reconoció al Sistema de Video Vigilancia Digilante como una ventaja para la XETID sirviendo de apoyo durante los servicios de guardia. A su vez se identificaron una serie de limitantes que dificultan el servicio del mismo.

Se propone para dar solución a las dificultades encontradas en el sistema, el desarrollo de una aplicación para dispositivos móviles con sistema operativo Android, que permita el monitoreo de las cámaras disponibles en el Sistema de Video Vigilancia Digilante. El entorno de despliegue de la solución debe contar con una red inalámbrica wifi con un alcance tal que desde cualquier lugar de la entidad sea posible la conexión con el servidor para el monitoreo de las cámaras. Para el uso de la aplicación el usuario debe autenticarse previamente y en dependencia del usuario que se autentique será el nivel de acceso que tenga a las cámaras disponibles en el sistema. Una vez obtenidas las *urls* (Localizador Uniforme de Recursos. *Uniform Resource Locator*, por sus siglas en inglés) de las cámaras disponibles tendrá la posibilidad de acoplarlas en una plantilla acorde a su conveniencia. Posteriormente el usuario podrá trabajar sobre las cámaras acopladas en la plantilla y si desea controlar alguna en específico deberá hacer *touch*[7] sobre su ubicación en la pantalla y podrá manipular la misma con las funcionalidades: hacer zoom, rotar campo de visión de la cámara, tomar fotos y grabar video de la cámara.

2.3 Definición de Requisitos

Los requisitos funcionales son declaraciones de los servicios que el sistema debe proporcionar, el cual se aplica a entradas particulares y situaciones donde debe mostrar correctamente los resultados deseados. Son condiciones o capacidades que debe cumplir el software, por lo que estas necesidades las plantean los clientes, evidenciando que debe tener y hacer el sistema.[33] A continuación, se detallan los requisitos funcionales y los no funcionales con el fin de lograr un producto estable y que cumpla con los objetivos planteados.

[7] Acción de tocar, en inglés (toque).

Requisitos funcionales

Los requisitos funcionales se caracterizan por ser parte fundamental para la creación de un software, pues se especifican a partir de las necesidades planteadas por el cliente. De ellos se desglosan las funcionalidades que se implementarán en el sistema.[34] En la tabla que se muestra a continuación se encuentran los requisitos funcionales definidos para el desarrollo de la aplicación.

Tabla 3. Requisitos funcionales de la aplicación

No	Nombre	Prioridad
RF1	Autenticar usuario.	Alta
RF2	Listar cámaras.	Alta
RF3	Seleccionar plantilla.	Media
RF4	Asignar cámara a la vista.	Baja
RF5	Eliminar cámara de la vista.	Baja
RF6	Rotar campo de visión de la cámara.	Alta

RF7	Guardar fotos.	Alta
RF8	Comenzar a grabar	Alta
RF9	Detener grabación	Alta
RF10	Ampliar imagen de la cámara.	Alta
RF11	Mostrar notificaciones de eventos.	Alta

Requisitos no funcionales

Los requisitos no funcionales son las restricciones de los servicios o funciones ofrecidas por el sistema, restringen el espacio de posibles soluciones. Debe pensarse en estos como las características que hacen al producto atractivo, usable, rápido o confiable.[35] A continuación se muestra la tabla que contiene los requisitos no funcionales definidos.

Tabla 4. Requisitos no funcionales de la aplicación

No	Nombre	Clasificación
RNF1	El usuario accederá a la aplicación mediante un sistema de autenticación.	Seguridad
RNF2	Usar como sistema operativo Android a partir de su versión 4.1.	Software
RNF3	Entrada por teclado o pantalla táctil.	Software
RNF4	Conectividad a una red wifi.	Hardware
RNF5	La aplicación debe ejecutarse en un dispositivo móvil con tamaño de pantalla	Hardware

	mínima de 480 x 800 píxeles y memoria RAM[8] de 512MB[9].	

2.4 Descripción de los requisitos funcionales

A continuación, se muestra la tabla del requisito funcional “Autenticar usuario”. Ver en **Anexo 2** para consultar tablas de descripción de requisitos funcionales.

Tabla 5. Autenticar usuario

Conceptos tratados	**Conceptos**	**Atributos**
	• Servicio web	• Usuario • Contraseña
Precondiciones	**Precondiciones**	**Pre-requisitos**
	• Debe estar conectado el dispositivo a la red. •	• Configurar la dirección del servidor web.

8 RAM, por sus siglas en inglés (Memoria de Acceso Aleatorio)

9 MB, por sus siglas en inglés (Megabyte). Unidad de medida de cantidad de datos informáticos.

Descripción	• Si el dispositivo está conectado a la red y la dirección del servidor web ha sido configurada correctamente el trabajador entra su usuario y contraseña y acciona el botón “Autenticar.” • Si el envío fue exitoso el servidor responde satisfactoriamente mediante un JSON. • Si hubo algún error en el proceso este es reportado mediante un JSON. • Si no ocurren errores en el proceso finaliza exitosamente.
Validaciones	• El campo contraseña y el campo usuario no deben estar vacíos.
Complejidad	• Media.
Post-condiciones	• El trabajador se autentica en el sistema.
Post-requisitos	• No procede.

2.5 Prototipos de interfaz de usuario

Los prototipos son una representación limitada de un producto que permite a las distintas partes probarlo en situaciones reales o explorar su uso, creando así un **proceso de diseño de iteración** que genera calidad. Son útiles para comunicar, discutir y definir ideas entre los **diseñadores** y las partes responsables. Apoyan la **evaluación de productos**, clarifican **requisitos de usuario** y definen alternativas. Se emplea el prototipo de baja fidelidad porque utiliza materiales distintos al del producto final y es más simple y fácil de producir, resultando ser particularmente útil durante el diseño conceptual. (Alberto Lacalle, “Prototipos”. julio de 2006)

A continuación, se muestran los prototipos de interfaz de usuario de la aplicación DigilanteDroid.

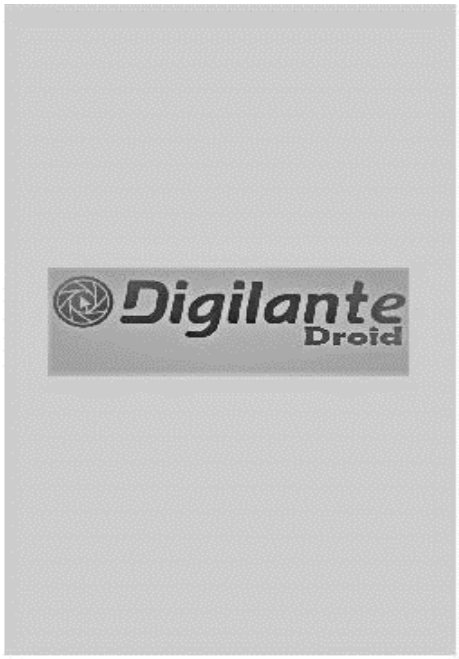


Figura 2. Splash de la aplicación

Figura 3. Login de la aplicación

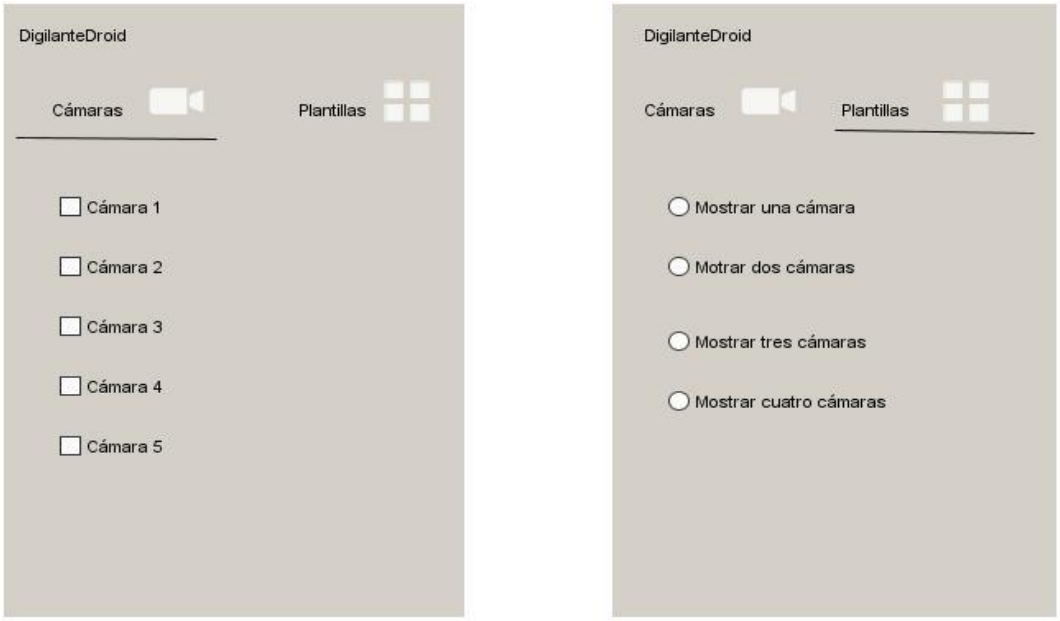


Figura 4. Listado de cámaras de la aplicación

2.6 Diseño de la arquitectura del sistema

La arquitectura empleada durante el desarrollo de la solución fue la arquitectura Modelo-Vista-Presentador, en adelante MVP, que surge del patrón de diseño Modelo Vista Controlador y propone la separación en Objetos Modelo, Objetos de Vista y Objetos Presentador. **[36]** A continuación se muestra un ejemplo de cómo se evidencia el patrón en la propuesta de solución.

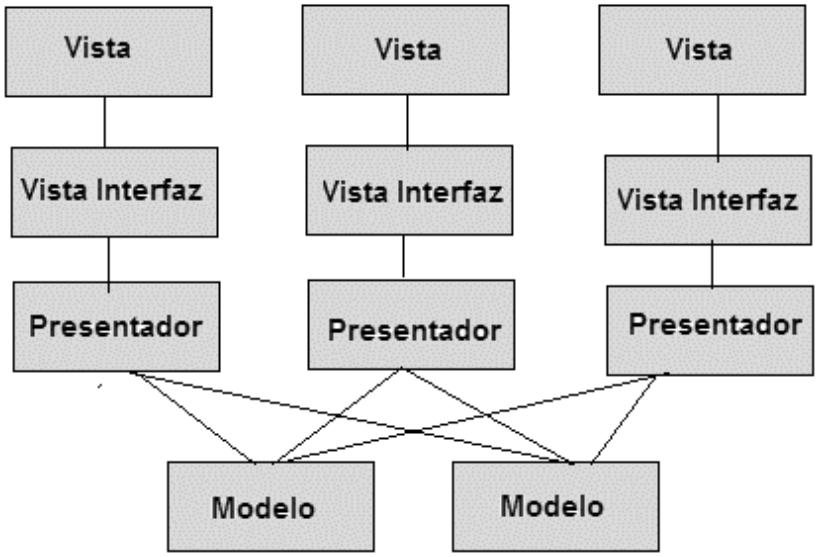


Figura 5. Modelo Vista Presentador. Fuente: Elaboración propia

Para seguir el patrón MVP se tienen en cuenta los siguientes puntos:

- El módulo de presentación, sólo trabajará con objetos Model, que contendrán únicamente la información necesaria para mostrarse en la Interfaz obteniendo, así, una capa de datos mucho más liviana y con los datos en memoria estrictamente necesarios. [36]
- Los objetos Vista, en Android, *Activities* y *Fragments*, recibirán datos y los mostrarán y avisarán al Presentador de los eventos que vayan sucediendo a lo largo de su vida. En ningún caso contendrán lógica. A su vez, estas Vistas se comunicarán con el Presentador a través de interfaces desconociendo, así, su implementación, consiguiendo desacoplar la Vista del Presentador, y permitiendo que el conjunto sea mucho más testeable. [36]
- Los objetos Presentador serán los encargados de estar a la escucha de los eventos ocurridos en los objetos Vista y realizar las acciones correspondientes para nutrir de datos a ésta. A su vez, éste formateará los datos que reciba para transformarlos a objetos de tipo Model y así conseguir el desacoplamiento entre la Interfaz y nuestra capa de datos propia. Será el encargado de notificar eventos a la Vista, ejemplo, cuando el Usuario utilice el botón Autenticar, la Vista utilizará el método

de la interfaz *LoginView*, el *Presenter* delegará en el Caso de Uso correspondiente y, al acabar, será el *Presenter* quien avise a la Vista de que debe dar *feedback* al Usuario.[36]

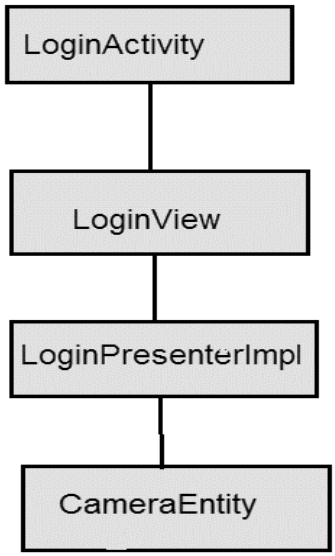


Figura 6. Ejemplo de Modelo Vista Presentador. Fuente: Elaboración propia

En la figura 6 se evidencia el empleo del patrón MVP en la aplicación desarrollada, en un ejemplo que muestra las distintas capas de la aplicación y como se relacionan entre sí. En la capa Modelo se encuentra la clase CameraEntity, entidad que contiene la información referente a las cámaras que se obtienen del servidor. En la capa Presentador, se muestra la clase LoginPresenterImpl que contiene la lógica necesaria para la autenticación del usuario al sistema. La interfaz LoginView encargada de la comunicación entre la vista y el presentador; y en la capa Vista se encuentra LoginActivity que es la clase vista encargada de mostrar la interfaz de autenticación del usuario.

2.7 Conclusiones del capítulo

Al concluir la modelación de la propuesta de solución, se alcanzan los siguientes resultados:

- Con la elaboración del modelo de dominio se obtuvo una perspectiva de los principales conceptos, obteniendo un mejor dominio y compresión del problema para identificar las posibles soluciones.
- Con la definición de los requisitos funcionales y no funcionales de la solución, se logró especificar las condiciones y restricciones que debe cumplir la aplicación, sirviendo de guía para la implementación del mismo.

- La utilización de la arquitectura MVP y la elaboración de prototipos de interfaz de usuario contribuyó al diseño de la aplicación, proporcionando una estructura para la misma y posibilitando el empleo de buenas prácticas de programación, así como el resultado de una primera vista de cómo debe quedar el sistema luego de la implementación.

Capítulo 3: Diseño de la aplicación informática para el monitoreo de cámaras del sistema de Video Vigilancia Digilante

En el presente capítulo se definen los modelos de implementación para desarrollar la aplicación informática que permitirá monitorear las cámaras del sistema de Video Vigilancia Digilante. Para ello se presentan los diagramas de clases del diseño, el diseño de la base de datos y se definen los patrones de diseño.

3.1 Diagrama de clases del diseño

El Diagrama de Clases del Diseño (DSD) describe gráficamente las especificaciones y las relaciones entre las clases del diseño. Se utiliza para modelar una vista de diseño estática de un sistema, modelar el vocabulario del sistema, modelar las colaboraciones o modelar esquemas. Por lo cual son importantes no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa [37]. A continuación, (ver Figura 5) se muestra el diagrama de clases del sistema.

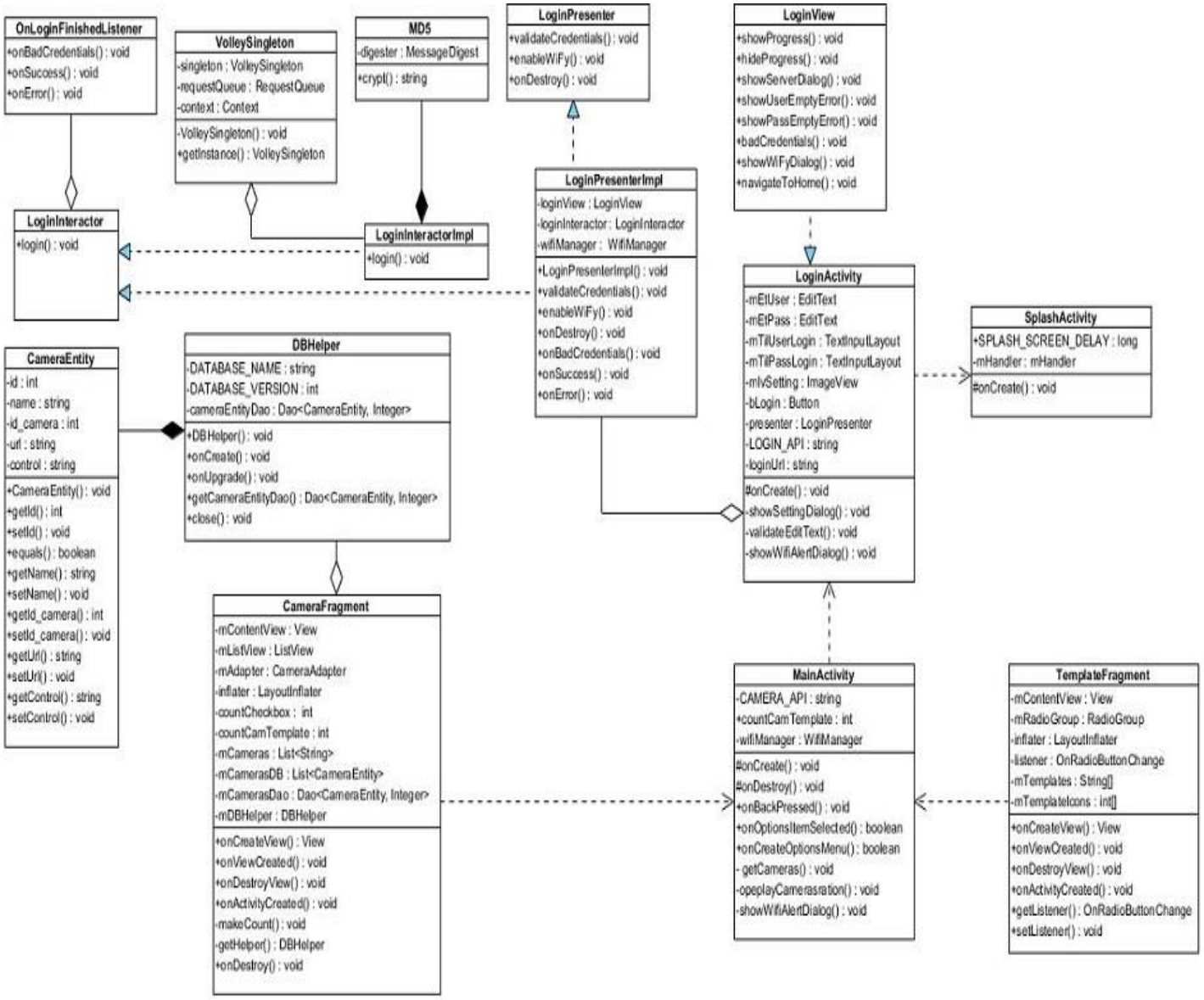


Figura 7. Diagrama de clases

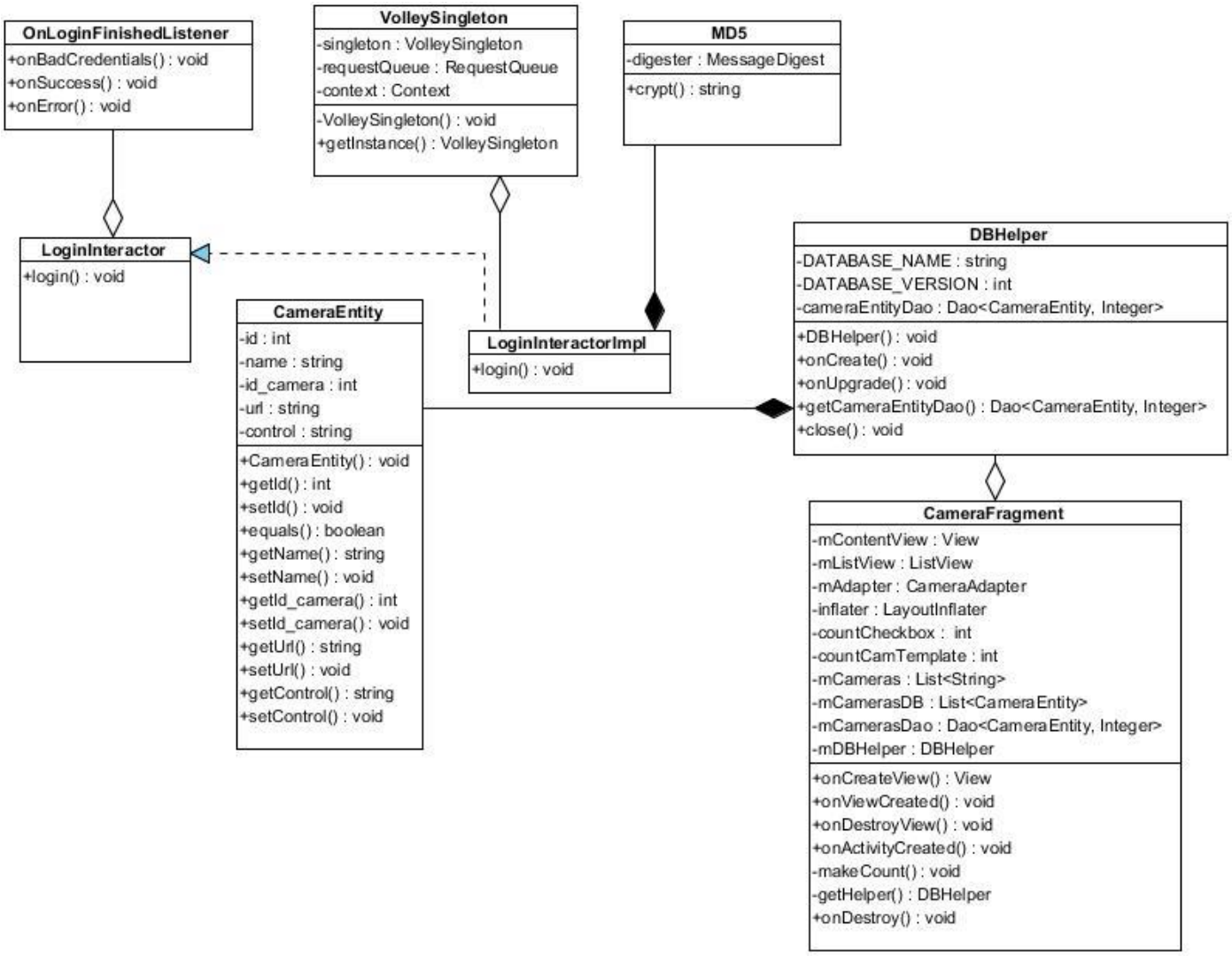


Figura 8. Diagrama de clases. Parte 1

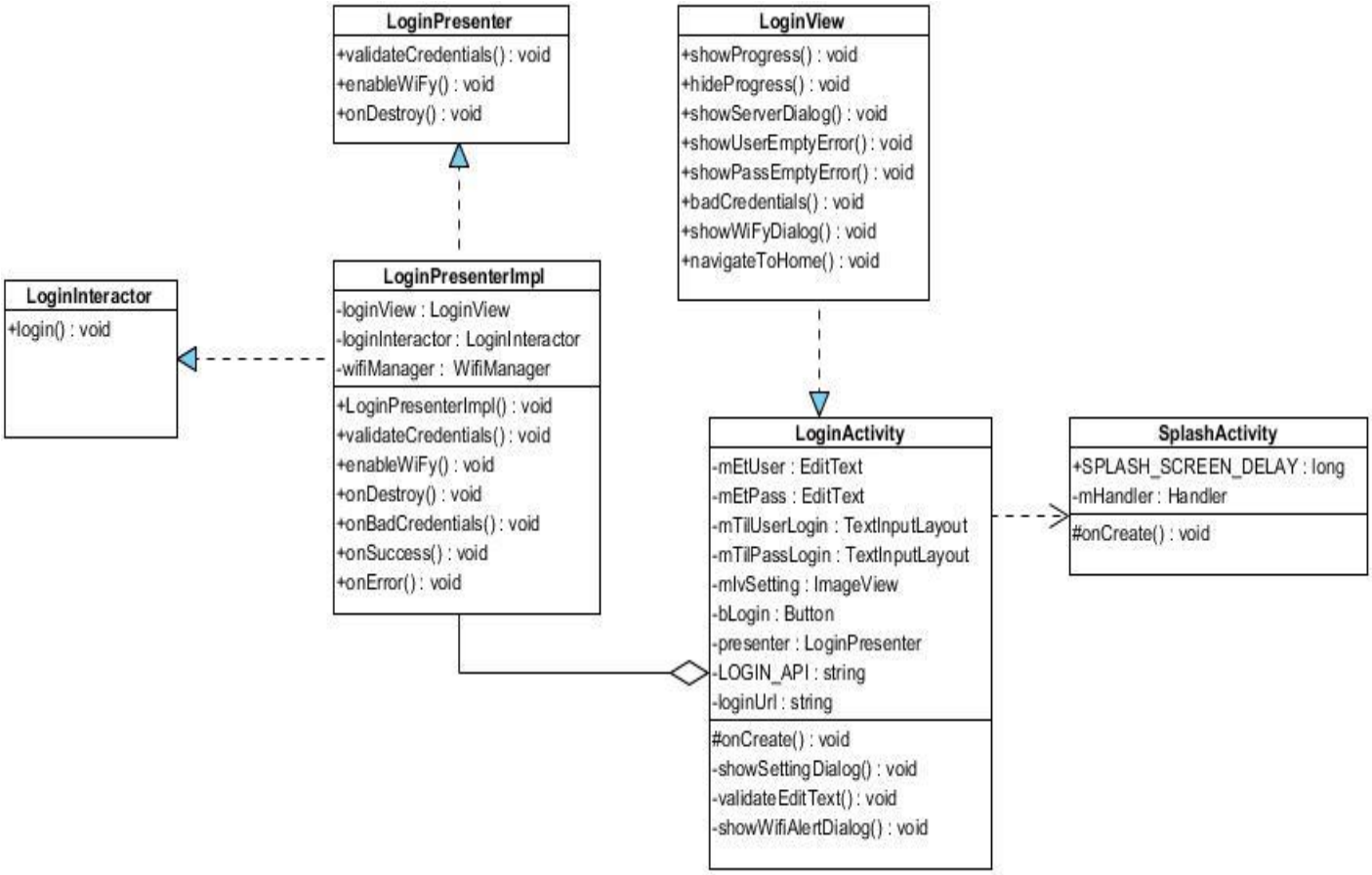


Figura 9. Diagrama de clases. Parte 2

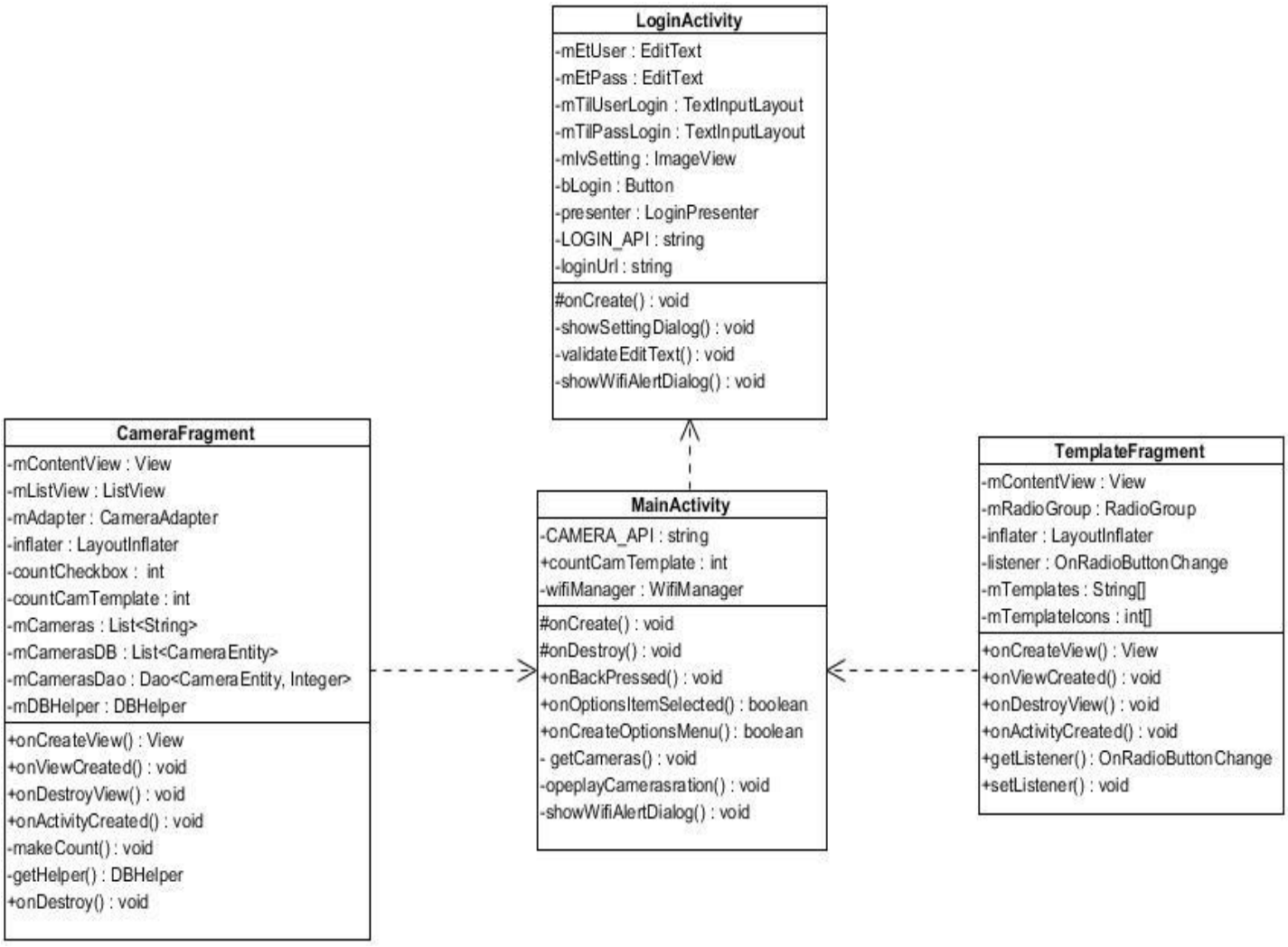


Figura 10. Diagrama de clases. Parte 3

3.2 Diseño de la base de datos

Un modelo de datos es un sistema formal y abstracto que permite describir los datos de acuerdo con reglas y convenios predefinidos. También se puede decir que es un conjunto de conceptos que permiten describir, a distintos niveles de abstracción, la estructura de una base de datos. **[38]** A continuación, la figura 7 muestra el diseño de la base de datos del sistema.

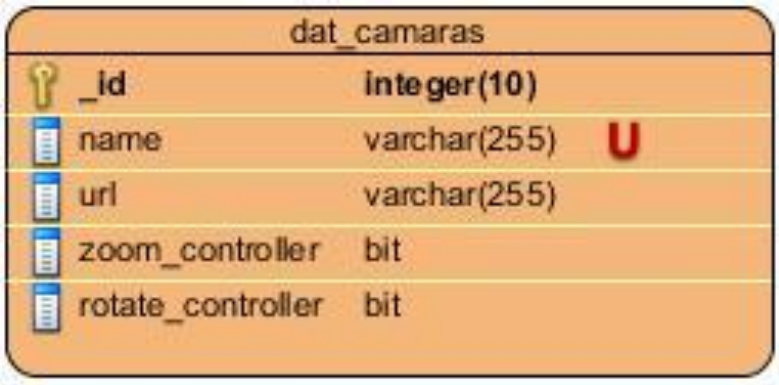


Figura 11. Diseño de la base de datos

El diseño de la base de datos muestra la única tabla que se utiliza de base d datos que se utiliza para almacenar las cámaras. La misma cuenta con los campos: (id, name, url, zoom_controller, rotate_controller).

3.3 Patrones de diseño

Patrones GRASP

Los Patrones para Asignar Responsabilidades (GRASP del inglés Responsability Assignment Patterns) describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño del objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. En el diseño de la aplicación se emplearon patrones de diseño GRASP tales como: [30]

- **Creador:** El patrón creador se pone de manifiesto en la clase *DBHelper* creando una instancia de la clase *BD* puesto que guía la asignación de responsabilidades relacionadas a la creación de objetos, una tarea muy común en sistemas orientados a objetos. Esto se aplica en todos los casos donde una clase tiene la responsabilidad de crear una nueva instancia de la otra. [30]

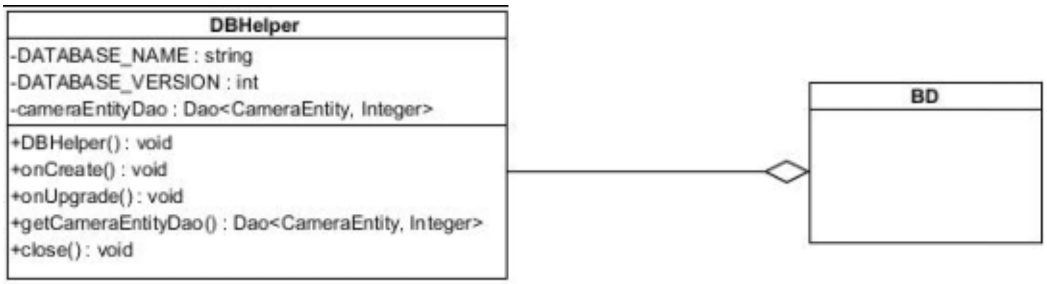


Figura 12. Patrón Creador. Fuente: elaboración propia

- **Experto:** El patrón experto en información soluciona el problema ¿de qué forma podemos saber qué responsabilidad delegar a cada objeto? Este principio básico de asignación de responsabilidades se manifiesta en MainActivity, que es la clase que domina la información necesaria para llevar a cabo la creación de un objeto o la implementación de un método. [30]

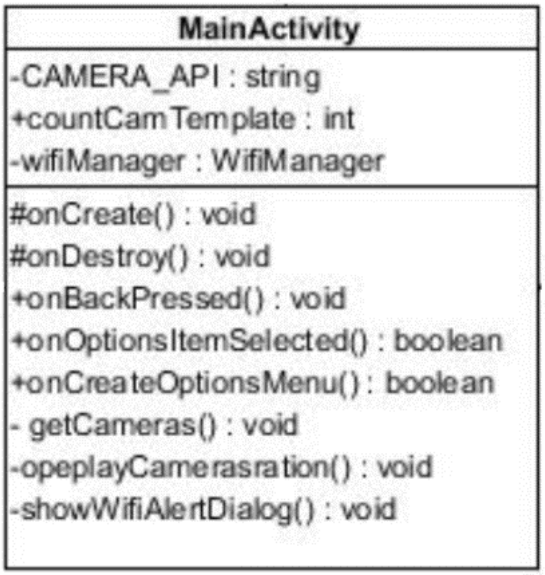


Figura 13. Patrón Experto. Fuente: elaboración propia

- **Controlador:** El patrón controlador se evidencia en las clases LoginPresenterImpl y LoginActivity sirviendo de intermediario entre la interfaz LoginActivity y el algoritmo LoginPresenterImpl de tal forma que la primera sea quien recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.[30]

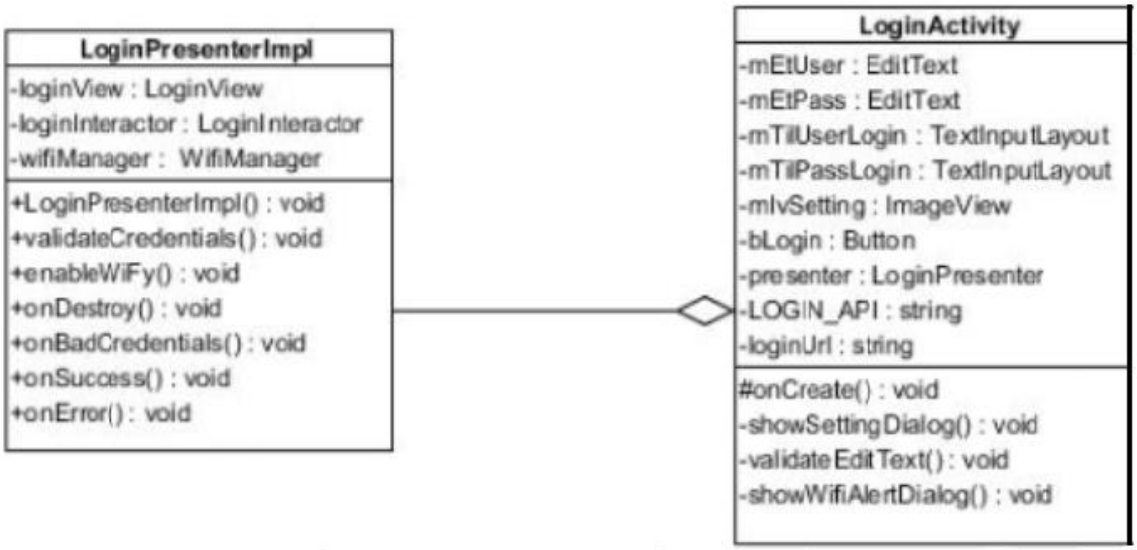


Figura 14. Patrón Controlador. Fuente: elaboración propia

- **Alta Cohesión:** Soluciona el problema de ¿cómo mantener manejable la complejidad? ya que asigna responsabilidades de manera que la información que almacena una clase sea coherente y esté relacionada con la clase. Este patrón se evidencia en todas las clases ya que ellas almacenan la información necesaria para trabajar, sin involucrar otras clases. [30]
- **Bajo Acoplamiento:** Soluciona el problema de ¿cómo dar soporte a las bajas dependencias y al incremento de la reutilización? Plantea tener las clases lo menos ligadas entre sí, de tal forma que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre las mismas. Este patrón se evidencia en todas las clases porque existe la mínima dependencia entre ellas para realizar modificaciones sin alterar su comportamiento. [30]

Patrones GOF

- **Singlenton:** Su objetivo es restringir la creación de objetos pertenecientes a una clase, de modo que solo se tenga una única instancia de la clase para toda la aplicación, garantizando así un punto de acceso global al objeto creado. Este patrón se evidencia en la clase VolleySingleton porque a través de esta clase se realiza el consumo de servicios web en la aplicación [30].
- Su objetivo es restringir la creación de objetos pertenecientes a la clase VolleySingleton de modo que solo se tenga una única instancia de la clase para toda la aplicación, lo que garantiza un punto

de acceso global al objeto creado y permite por ende la realización del consumo de servicios web [30].

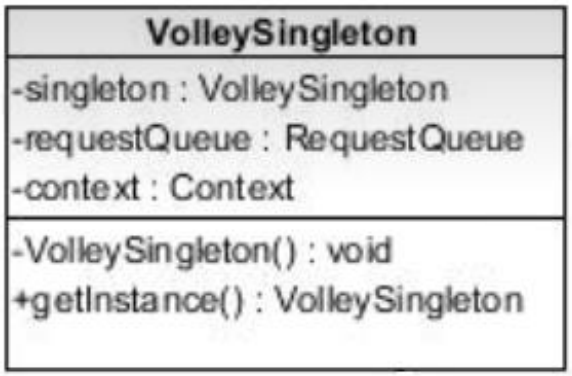


Figura 15. Clase VolleySingleton

3.4 Diagrama de Paquete

El empaquetado de la aplicación se ha representado en el diagrama que se muestra a continuación, donde se agrupan elementos relacionados semánticamente y se muestran las dependencias entre dichas agrupaciones. [30]

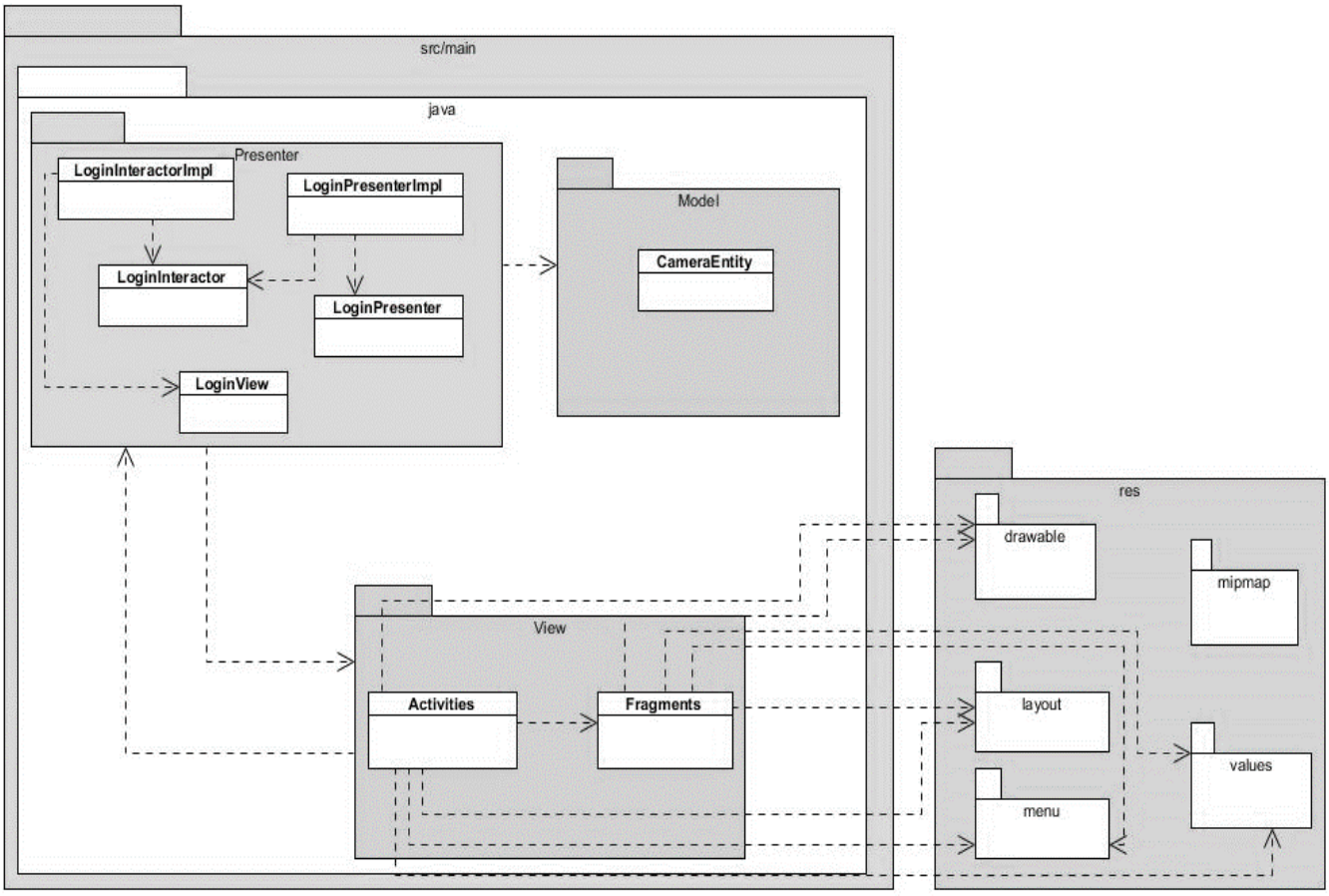


Figura 16. Diagrama de Paquete

La estructura de paquetes y archivos de un proyecto de este tipo es la siguiente: [30]

- **src/:** Contiene los archivos de clases y actividades. Estos son almacenados en la dirección: src/your/package/namespace/ActivityName.Java.
- **bin/:** Directorio de salida de la construcción del archivo.apk y otros recursos compilados.
- **res/:** Contiene los recursos de la aplicación, como archivos *drawable*, *layouts* y los valores *string* (archivo XML que contiene los textos visualizados en la herramienta).
- **drawable/:** Para archivos de imágenes.png, .jpeg o .gif; archivos XML que describen las formas *drawable* u objetos *drawable* que contengan múltiples estados.
- **layout/:** Archivos XML que son compilados en los *layouts* de pantalla.

- **values/:** Para archivos XML que son compilados en diversos tipos de recursos. A diferencia de otros recursos del directorio res/, los recursos que son escritos a archivos XML en esta carpeta no son referenciados por su nombre, sino que el tipo de elemento XML contenido es controlado a través de la clase R.
- **libs/:** Contiene las librerías privadas que son utilizadas internamente en el proyecto.

3.5 Conclusiones del capítulo

Luego de realizar el diseño del sistema se concluye que:

- Los patrones de diseño empleados garantizaron mantener un orden lógico entre las clases; lo cual posibilitó organizar el desarrollo de la aplicación.
- El diagrama de clase de diseño posibilitó, describir las especificaciones de las clases de software, obteniendo una representación técnica del sistema implementado.
- Definir la estructura de la base de datos del sistema, permitió puntualizar las principales fuentes de datos para dar soporte a las actividades de la entidad.

Capítulo 4: Implementación y evaluación de la aplicación informática para el monitoreo de cámaras del sistema de video vigilancia Digilante

En el presente capítulo se muestran los resultados de la implementación de los diferentes componentes de la aplicación, en términos de artefactos como el diagrama de despliegue y el diagrama de componentes, pertenecientes al flujo de trabajo en cuestión. Posteriormente se realizan pruebas al sistema con el propósito de identificar sus debilidades, comprobando que el producto final cumple con los requisitos establecidos.

4.1 Diagrama de componentes

El diagrama de componentes permite describir los elementos físicos del sistema y sus relaciones. Los componentes representan todos los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas, pueden ser simples archivos, paquetes o bibliotecas cargadas dinámicamente. Debido a que la complejidad de una aplicación se suele dividir por partes, representándose la estructura a alto nivel de los diferentes subsistemas y sus relaciones que intervienen en la solución de una funcionalidad **[37]**. Permite además una visión y organización real de los componentes de software que integran la solución en el ambiente de desarrollo, mostrando las relaciones y dependencias de cada uno de ellos.

El diagrama de componentes que muestra la figura 17 permite visualizar el comportamiento del servicio que proporciona y usa la aplicación a través de las interfaces. MainActivity representa la interfaz principal y contienen los Fragments: CameraFragment que brinda el listado de cámaras disponibles en el sistema una vez autenticado el usuario y TemplateFragment que ofrece una lista de plantillas en las que se acoplan las cámaras seleccionadas por el usuario.

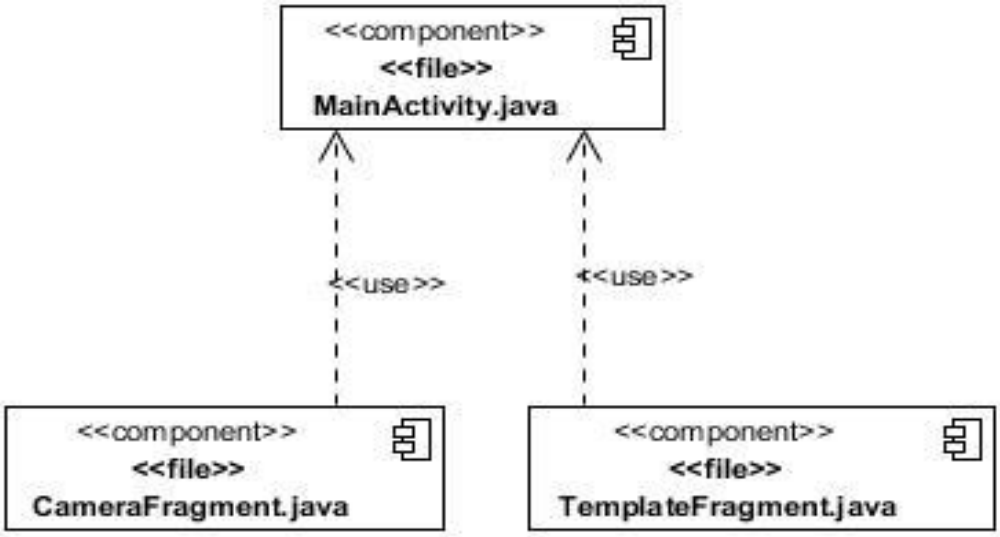


Figura 17. Diagrama de componentes

4.2 Diagrama de despliegue

El diagrama de despliegue es un tipo de diagrama UML que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño. Es útil para modelar el hardware empleado en las implementaciones de sistemas y las relaciones entre sus componentes, lo cual representa la capa estructural que define el comportamiento de los diversos dispositivos de hardware y lenguaje de programación.**[37]** En la Figura 18 se muestra el diagrama de despliegue para el sistema.

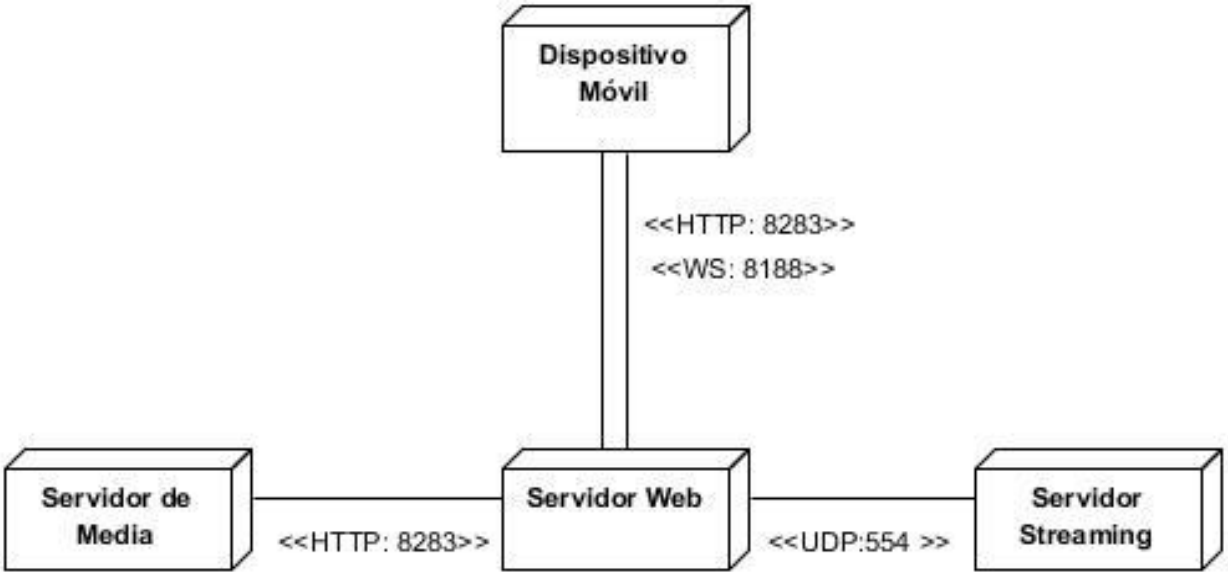


Figura 18. Diagrama de Despliegue

El diagrama representa la comunicación que existe entre los nodos que interactúan en el flujo de la información. Donde la aplicación y el servidor web, hacen uso del protocolo HTTP por el puerto 8283, para el proceso de autenticación y del protocolo WS por el puerto 8188, para el flujo de video.

La comunicación entre el servidor web y el servidor *streaming* se realiza a través del protocolo UDP por el puerto 554. Mientras que la comunicación entre el servidor media y el servidor web para almacenar fotos y videos grabados en la aplicación, se realiza a través el protocolo HTTP por el puerto 8283.

4.3 Pruebas del Software

Las pruebas de software se realizan con la finalidad de descubrir errores, tanto en la lógica interna como en funciones externas del mismo. Son un elemento crítico para garantizar la calidad del software, permitiendo comprobar el correcto funcionamiento y las posibles respuestas que el producto despliega ante determinadas situaciones. Son básicamente un conjunto de actividades dentro del desarrollo de software, dependiendo del tipo, podrán ser implementadas en cualquier momento del proceso de desarrollo.**[39]**

Prueba de Sistema

Las pruebas de sistema se realizan una vez integrados todos los componentes de la aplicación. Su objetivo es ver la respuesta del sistema en su conjunto, frente a distintas situaciones. Se simulan varias alternativas

que podrían darse con el sistema implantado y en base a ellas se prueba la eficacia y eficiencia de la respuesta que se obtiene. Se pueden distinguir varios tipos de prueba distintos, por ejemplo: **[40]**

- Pruebas negativas: se trata de que el sistema falle para ver sus debilidades.
- Pruebas positivas: se trata de entrar valores al sistema para que devuelva un resultado esperado.

Método de Pruebas

El método de pruebas a realizar para las pruebas de sistema es el de Caja Negra. Este tipo de pruebas, conocidas también como pruebas funcionales o pruebas de comportamiento, concentran la atención en generar casos de prueba que permitan ejercitar los requisitos funcionales de un programa. A diferencia de las pruebas de caja blanca, que se basan en la lógica interna del software, este tipo de pruebas se concentran en su funcionalidad, por lo que mucho del trabajo se realiza interactuando con la interfaz del software. Los casos de prueba generados en este enfoque, se diseñan a partir de valores entrada y salida. De esta forma, se puede determinar la validez de una salida para un conjunto de entradas proporcionadas. La aplicación de pruebas de caja negra permite detectar errores como funciones incorrectas o ausentes, errores en estructuras de datos, errores de rendimiento, así como errores de inicialización y terminación. **[41]**, **[42]**, **[43]**

Técnica de Pruebas.

Para diseñar los casos de prueba se hará uso de la técnica de Partición Equivalente. La idea de esta técnica, es en dividir los valores válidos y no válidos para entradas y salidas en un número reducido de particiones de forma que, el comportamiento del software sea el mismo para cualquier valor contenido en una partición particular. El propósito principal de una partición es reducir la cantidad de casos de prueba generados en el proceso. **[41]**, **[42]**, **[43]**

Diseño del Caso de Prueba del Requisito Funcional "Autenticar usuario":

- **Condiciones de ejecución:**
  1. El dispositivo debe estar conectado a la red.
  2. La dirección del servidor web debe estar configurada correctamente.
  3. El trabajador debe introducir su usuario y contraseña.

Tabla 6. Requisito Funcional “Autenticar usuario”

Escenario	**Descripción**	**Usuario**	**Contraseña**	**Respuesta del Sistema**	**Flujo Central**
EP 1.1: Autenticar usuario introduciendo valores correctos.	El sistema debe permitir la autenticación del usuario.	V(administrador)	V(administrador)	1. Se muestra la interfaz Autenticar usuario. 3. El sistema le notifica mostrando el mensaje: “Acceso concedido ”	2.El usuario entra los parámetros de autenticación y da click en el botón "Autenticar”
EP 1.2: Autenticar usuario introduciendo valores incorrectos.	El sistema no debe permitir la autenticación del usuario.	V(administrador)	V(admin)	1. Se muestra la interfaz Autenticar usuario. 3. El sistema le notifica mostrando el mensaje: *“*Acceso denegado*”*.	2.El usuario entra los parámetros incorrectos y da click en el botón "Autenticar”
		V(admin)	V(administrador)		
		V(admin)	V(admin)		
		V(administrador)	V()		
		V()	V()		

- **Descripción de la variable**

Tabla 7. Descripción de la variable

No	Nombre de campo	Clasificación	Puede ser nulo	Descripción
1	Usuario	EditText	No	Texto
2	Contraseña	EditText	No	Caracteres

Resultado de las pruebas realizadas

Para la ejecución de la técnica Partición Equivalente perteneciente al método de Caja Negra se realizó un diseño de casos de pruebas. Obteniéndose un total de 2 escenarios de pruebas, los cuales arrojaron los siguientes resultados:

Para la 1era iteración se obtuvieron 4 no conformidades las mismas se muestran en la siguiente tabla:

Tabla 8. Resultados de prueba 1era iteración

No.	Descripción	CP[10]	Clasificación
1	El sistema no permitía agregar más de dos cámaras a la plantilla.	# 2	media
2	El sistema permitía seleccionar más cámaras que las admitidas en la plantilla.	# 2	baja

[10] Casos de Prueba.

3	Una vez adicionadas las cámaras a la vista el sistema no permitía eliminar las mismas	# 2	baja
4	El sistema mostraba la vista mostrar cámaras sin haber agregado ninguna.	# 2	media

Para la 2da iteración se obtuvo 1 no conformidad la misma se muestra en la siguiente tabla:

Tabla 9. Resultados de la segunda iteración

No.	Descripción	CP	Clasificación
1	Una vez seleccionadas las cámaras el sistema solo muestra la primera en la vista.	# 2	alta

Para la 3ra iteración no se obtuvieron no conformidades.

Ver en **Anexo 3** Diseño del caso de prueba del requisito funcional "Listar cámaras"**.**

Prueba de eficiencia

La prueba de eficiencia se realiza con el objetivo de evaluar el rendimiento de la aplicación en condiciones de uso habitual.[40] Ver en Anexo 4 pruebas de eficiencia realizadas a la aplicación utilizando la herramienta JMeter, mediante la cual se prueba la eficiencia al servicio más crítico de la aplicación: stop. Varios grupos de usuario hacen peticiones al servidor y obtienen respuesta en un período de tiempo comprendido en 3000 milisegundos (ms).

Los resultados de las pruebas arrojan que para una muestra 50 usuarios el tiempo de muestra es de 54 ms, para una media de 58 y error al 0%. De una muestra de 100 usuarios se obtiene un tiempo de muestra de 177 ms para una media de 99 y error al 0%. Una muestra de 150 usuarios lanza un tiempo de muestra de 300 ms y una media de 230 con un 0% de error; y para un total de 200 usuarios el sistema devuelve un tiempo de muestra de 3012 ms, sobrepasando el tiempo máximo de espera y por consiguiente arrojando una serie de errores con una media de 641.

Luego de realizar la prueba de rendimiento al sistema, se determina que el mismo admite hasta un total de 150 peticiones al servidor de manera simultánea ya que para una un grupo de 200 usuarios no se obtiene respuesta satisfactoria del servidor al sistema.

Pruebas de aceptación

Según, el ISTQB26 (*International Software Testing Qualification Board*, por sus siglas en inglés) define la "Aceptación" como: Pruebas formales con respecto a las necesidades del usuario, requerimientos y procesos de negocio, realizadas para determinar si un sistema satisface los criterios de aceptación que permitan que el usuario, cliente u otra entidad autorizada pueda determinar si acepta o no el sistema. **[44]**

Las pruebas de aceptación se clasifican en dos tipos: pruebas Alfa y pruebas Beta. El tipo de prueba aplicado fue pruebas Alfa, que se realizan por un cliente en un entorno controlado por el equipo de desarrollo. Para que tengan validez, se debe primero crear un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente. Una vez logrado esto, se procede a realizar las pruebas y a documentar los resultados. Cuando el software sea la adaptación de una versión previa, deberán probarse también los procesos de transformación de datos y actualización de archivos de todo tipo.**[40]**

El diseño de casos de pruebas de aceptación de usuario se define incluyendo como principal escenario de la aplicación el requisito funcional Listar cámaras.

La siguiente tabla muestra las pruebas de aceptación del RF2 perteneciente a la primera iteración de sistema. La cual fue escogida por ser una de los procesos más relevantes en el desarrollo del sistema.

Tabla 10. Caso de prueba Listar cámaras. Fuente: Elaboración Propia

Caso de Prueba Aceptación
Nombre: Mostrar cámaras
Nombre de la persona que realiza la prueba: Jorge Arce Martínez.
Descripción de la Prueba: Comprobar que se muestren las cámaras disponibles en el sistema teniendo en cuenta la plantilla seleccionada.
Condiciones de Ejecución: Conexión a una red *wifi*, configuración del servidor web y autenticación del usuario.
Entradas/ Pasos de Ejecución: - Seleccionar una plantilla con la cantidad de cámaras que desea visualizar el usuario. - Agregar las cámaras acordes a la plantilla seleccionada. - Accionar el botón **play** para mostrar las cámaras.
Resultado esperado: El usuario visualiza las cámaras seleccionadas.
Evaluación de la prueba: Prueba satisfactoria

Tabla 11. Resultados de prueba

Pruebas del sistema	**RF**	**Iteración**	**NC**	**Cerrada**	**No Procede**
	2	1ra	2	2	0
		2da	0	0	0

Una vez realizados el caso de Prueba para el requisito funcional Listar cámaras, con dos iteraciones se eliminaron las 2 No Conformidades.

Como resultado se detectó que los errores encontrados pueden clasificarse según su tipo:

- Para la primera No Conformidad se revelan errores de Presentación porque no se visualiza correctamente toda la información que se desea mostrar.
- Para la segunda No conformidad el error detectado fue de Ejecución, durante el despliegue del sistema se detectó una falla, ocasionada porque el sistema no realizaba todos los procedimientos definidos.

4.4 Conclusiones del capítulo

El desarrollo del presente capítulo, permitió realizar la implementación y las pruebas del sistema, arrojando como resultados:

- El diagrama de componente del sistema posibilitó obtener una visión y organización real de las interfaces que integran la aplicación.
- La realización del diagrama de despliegue ofreció una vista para modelar el hardware empleado en las implementaciones de sistemas y las relaciones entre sus componentes.
- Realizar pruebas a la aplicación posibilitó verificar y validar el funcionamiento del sistema, obteniendo resultados satisfactorios, ya que los errores encontrados fueron corregidos.

Conclusiones

A partir de la investigación realizada y los resultados obtenidos, se concluye lo siguiente:

1. El estudio de las tecnologías y sistemas homólogos permitió establecer las bases teóricas e identificar las principales características de las aplicaciones para monitoreo y control de cámaras de video vigilancia para el posterior desarrollo de la solución.
2. El diseño de la solución propuesta permitió obtener una vista global de su estructura, sirviendo de base para comenzar su implementación.
3. Se desarrolló una aplicación informática que permite el monitoreo de las cámaras del Sistema de Video Vigilancia Digilante que responde a los requisitos definidos y al problema de investigación planteado.
4. La evaluación de la propuesta de solución a través de pruebas demuestra que la misma cumple con las funcionalidades exigidas y está apta para ser desplegada en un entorno real.

Recomendaciones

1. La aplicación para una nueva versión debería contar con un almacenamiento en un directorio local del dispositivo de forma que coincida con la información almacenada en el servidor de media del sistema de video vigilancia.

Bibliografía

[1] Cat Colombia Solutions, «LA EVOLUCIÓN DE LOS SISTEMAS DE SEGURIDAD ELECTRÓNICA». 2016.

[2] V. Lio, «CIUDADES,CÁMARAS DE SEGURIDAD Y VIDEO VIGILANCIA: ESTADO DEL ARTE Y PERSPECTIVAS DE INVESTIGACIÓN». .

[3] Grupo Eurofesa, «La videovigilancia, uno de los sistemas de seguridad más utilizados del mundo». mar-2017.

[4] Revista Cubana de Ciencias Informáticas, «Seguimiento de múltiples personas en aplicación de videovigilancia con cámaras de baja resolución». abril-junio 20017.

[5] S. Rodríguez Vázquez, Y. Tirado Gutiérrez, y G. Lozuna Farías, «Subsistema Visor APK». .

[6] I. E. Greenberg, 09-sep-2010.

[7] F. Klauser, 2004.

[8] A. Del Rey Abad, «Componente video-sensor de detección de fuego para el Sistema de Video-Vigilancia Xilema Suria.» jun-2015.

[9] Ayuntamiento de Gijón, «Guía de negocios: Sistemas de videovigilancia.» .

[10] R. Morales Toledo y S. Caballero Sarduy, «Multiplexor de flujos de video provenientes de cámaras IP». 2015.

[11] J. R. Jova Rodríguez, «Módulo de streaming para archivos multimedia del Sistema Integral de Análisis de Información». 2013.

[12] A. Fumero, G. Roca, y F. Sáez Vaca, «Diseño de materiales multimedia. Web 2.0». .

[13] J. González Pisano, «REST – REpresentational State Transfer». 2007-2006.

[14] M. P. Papazoglou, «Web Services: Principles & Technology». 2008.

[15] «Curso librerías Web 2.0». nov-2008.

[16] D. Sanz, M. Saucedo, y P. Torralbo, «Introducción a Android». E.M.E. Editorial.

[17] Y. Gómez Arteaga, «Estudio de estructura de Streaming de Video en Sistemas Móviles Celulares más allá de Tercera Generación (B3G), posible aplicación en Cuba». 29-jun-2011.

[18] «Tecnología Wi-Fi». .

[19] D. M. Costilla y S. Reaño Montoro, «Streaming de Audio/Video». 2008-2007.

[20] D. D. Morató, «Redes de Ordenadores. Nivel de transporte: UDP». .

[21] M. D. Banchoff Tzancof, «Websocket: Comparación de performace e iplemenración de aplicaciones web.» abril-2011.

[22] S. l Rodríguez Vázquez, Y. Tirado Gutiérrez, y G. Luzua Farías, «Sistema de Video Vigilancia Xilema Suria para la protección y seguridad de entidades. Subsistema Visor apk». .

[23] D. G. Mateo, «Las mejores apps de videovigilancia.» may-2013.

[24] «IP Cam Viewer». .

[25] «TinyCam Monitor». .

[26] UCID, «Proceso de Desarrollo y Gestión de Proyectos de Software (Versión 1.5)». 2012.

[27] J. Milanés Shelton y S. Leyva Ramire, «Herramienta de colaboración empresarial para gestionar el conocimiento mediante redes sociales». jun-2017.

[28] Jiménez Rodríguez, «Plataforma para la Gestión de Juegos Multiusuario de Teléfonos Móviles vía GPRS». .

[29] T. Groussard, «Java 7. Los fundamentos de lenguaje Java.» Ediciones ENI, mar-2012.

[30] E. Salabarria Aquino, «Aplicación Android para apoyar la estrategia de respuesta ante incidentes de peligrosidad en Cuba». 2017.

[31] Gray Watson, «OrmLite - Lightweight Object Relational Mapping (ORM) Java Package». jul-2016.

[32] J. L. Serradilla, «Control de Versiones con Sobversion TortoiseSVN». 2007.

[33] I. Somerville, «Ingeniería de Software». 2005.

[34] A. Del Rey Abad, «Componente video-sensor de detección de fuego para el Sistema de Video-Vigilancia Xilema Suria.» jun-2015.

[35] R. S. Pressman, «“Software Engineering”.» 2010.

[36] M. González Díez, «CLEAN ARCHITECTURE Y RXJAVA EN ANDROID». 30-jun-2016.

[37] I. Jacobson, «El Proceso Unificado de Desarrollo de Software.» 2000.

[38] M. REDONDO, «Estudio con modelos de datos para la automatización en redes eléctricas inteligentes.» 31-may-2017.

[39] R. S. Pressman, «Ingeniería de Software, Un enfoque práctico.» Madrid: 5ta edición, 2001.

[40] P. Suárez y C. Fontela, «Documentación y pruebas antes del paradigma de objetos.» 2003.

[41] R. Pressman, «Ingeniería del software. Un enfoque práctico». McGraw-Hill/Interamericana de España, 1998.

[42] D. C. Jorgensen, «Software testing. A craftsman’ s approach». 1995.

[43] E. Kit, «Software testing in the real world. Improving the procces». 1996.

[44] PMOinformatica La oficina de proyectos de informática, «Pruebas de aceptación de software según el ISTQB.» ago-2016.

Anexos

Anexo 1 Entrevista

Tabla 12. Entrevista realizada al Ing. Luis Ramón Samada de la Paz, especialista en temas de video vigilancia del centro de Tecnologías de SCADA y Seguridad.

No.	Preguntas realizadas
1	¿Qué mecanismo de comunicación provee el Sistema Digilante?
2	¿Cómo se maneja la seguridad del sistema?
3	¿Qué protocolo utiliza el sistema para publicar el flujo de video de las cámaras?
4	¿Cuáles son los roles definidos en el Sistema Digilante?

Anexo 2. Descripción de Requisitos funcionales

Tabla 13. Listar cámaras

Conceptos tratados	**Conceptos**	**Atributos**
	• Servicio web • JSON	• id_usuario
Precondiciones	**Precondiciones**	**Pre-requisitos**
	• Debe estar conectado el dispositivo a la red.	• Debe estar autenticado el usuario.

Descripción	• Si el dispositivo está conectado a la red y el usuario está autenticado el usuario acciona el botón  . • Si el envío de la petición al servidor fue exitoso se procede a parsear la respuesta del servidor. • Si la respuesta fue positiva el servidor devuelve el listado de las cámaras disponibles.
Validaciones	• No procede.
Complejidad	• Media.
Post-condiciones	• No procede.
Post-requisitos	• No procede.

Tabla 14. Seleccionar plantilla

Conceptos tratados	**Conceptos**	**Atributos**
	Plantilla	• cant_camera
Precondiciones	**Precondiciones**	**Pre-requisitos**
	• Debe existir un listado de plantillas.	• Debe estar autenticado el usuario.
Descripción	• Si existe un listado de plantillas y el usuario está autenticado, debe seleccionar la plantilla que estime conveniente accionando en el RadioButton correspondiente a su selección.	
Validaciones	• No procede.	

Complejidad	• Media.
Post-condiciones	• No procede.
Post-requisitos	• No procede.

Tabla 15. Asignar cámara a la vista

Conceptos tratados	**Conceptos**	**Atributos**
	Cámara	• plantilla • id_camera • name
Precondiciones	**Precondiciones**	**Pre-requisitos**
	• Debe existir un listado de cámaras disponibles.	• Debe estar seleccionada la plantilla para la vista.
Descripción	• Si existe un listado de cámaras disponibles y se ha seleccionado una plantilla, entonces se deben accionar los CheckBox correspondientes a las cámaras que se deseen asignar a la vista, de acuerdo a la plantilla seleccionada. • Si el usuario ha seleccionado una plantilla pero no existe un listado de cámaras, entonces no se puede mostrar la vista.	
Validaciones	• No procede.	
Complejidad	• Media.	
Post-condiciones	• No procede.	
Post-requisitos	• No procede.	

Tabla 16. Eliminar cámara de la vista

Conceptos tratados	**Conceptos**	**Atributos**
	Cámara	• Plantilla • Id_camara • name_camara
Precondiciones	**Precondiciones**	**Pre-requisitos**
	• Debe existir un listado de cámaras disponibles.	• Debe estar seleccionada la plantilla para la vista.
Descripción	• Si existen cámaras asignadas a la vista entonces se debe accionar en el CheckBox correspondientes a la cámara que se desee eliminar. • Si no existen cámaras asignadas a la vista entonces no se puede eliminar cámara.	
Validaciones	• No procede.	
Complejidad	• Media.	
Post-condiciones	• No procede.	
Post-requisitos	• No procede.	

Tabla 17. Comenzar a grabar

Conceptos tratados	**Conceptos**	**Atributos**
	• *Streamming*	• No procede
Precondiciones	**Precondiciones**	**Pre-requisitos**

	• Debe estar conectado el dispositivo a la red. • Servidor disponible.	• Obtención del flujo de video de la cámara.
Descripción	• Si el servidor está disponible se captura el flujo de video de la cámara y se guarda en un directorio previamente configurado. • Si el servidor no está disponible se muestra un mensaje de error. • Si ocurre un error al guardar la instantánea se muestra un mensaje de error. • Si no ocurren errores el proceso finaliza exitosamente.	
Validaciones	• No procede.	
Complejidad	• Alta.	
Post-condiciones	• No procede.	
Post-requisitos	• No procede.	

Tabla 18. Detener grabación

Conceptos tratados	**Conceptos**	**Atributos**
	• *Streamming*	• No procede
Precondiciones	**Precondiciones**	**Pre-requisitos**

	• Debe estar conectado el dispositivo a la red. • Servidor disponible.	• Obtención del flujo de video de la cámara.
Descripción	• Si el servidor está disponible se captura el flujo de video de la cámara y se guarda en un directorio previamente configurado. • Si el servidor no está disponible se muestra un mensaje de error. • Si ocurre un error al guardar la instantánea se muestra un mensaje de error. • Si no ocurren errores el proceso finaliza exitosamente.	
Validaciones	• No procede.	
Complejidad	• Alta.	
Post-condiciones	• No procede.	
Post-requisitos	• No procede.	

Tabla 19. Rotar campo de visión de la cámara

Conceptos tratados	**Conceptos**	**Atributos**
	• *Servicio web* • *JSON*	• id_camara
Precondiciones	**Precondiciones**	**Pre-requisitos**

	• La cámara debe tener un controlador de movimiento.	• Detención de la cámara.
Descripción	• Si el servidor está disponible se envía la petición de rotar la cámara. • Si el servidor no está disponible se muestra un mensaje de error. • Si no ocurren errores el proceso finaliza exitosamente.	
Validaciones	• No procede.	
Complejidad	• Media.	
Post-condiciones	• La cámara modifica el ángulo de visión.	
Post-requisitos	• Detención de movimiento.	

Tabla 20. Guardar fotos

Conceptos tratados	**Conceptos**	**Atributos**
	• No procede	• No procede
Precondiciones	**Precondiciones**	**Pre-requisitos**

	• Debe estar conectado el dispositivo a la red. • Servidor disponible.	• Obtención del flujo de video de la cámara.
Descripción	• Si el servidor está disponible se toma una instantánea del video de la cámara y se guarda en un directorio previamente configurado. • Si el servidor no está disponible se muestra un mensaje de error. • Si ocurre un error al guardar la instantánea se muestra un mensaje de error. • Si no ocurren errores el proceso finaliza exitosamente.	
Validaciones	• No procede.	
Complejidad	• Media.	
Post-condiciones	• No procede.	
Post-requisitos	• No procede.	

Tabla 21. Ampliar imagen de la cámara

Conceptos tratados	**Conceptos**	**Atributos**
	• Servicio web • JSON	• Id_camara
Precondiciones	**Precondiciones**	**Pre-requisitos**

	• La cámara debe tener un controlador de Zoom.	• Detención de movimiento.
Descripción	• Si el servidor está disponible se envía la petición de aumentar o disminuir el factor de Zoom. • Si el servidor no está disponible se muestra un mensaje de error. • Si no ocurren errores el proceso finaliza exitosamente.	
Validaciones	• No procede.	
Complejidad	• Media.	
Post-condiciones	• La cámara modifica su factor de Zoom.	
Post-requisitos	• No procede.	

Tabla 22. Mostrar notificaciones de eventos

Conceptos tratados	**Conceptos**	**Atributos**
	• Servicio web • JSON	• Id_camara
Precondiciones	**Precondiciones**	**Pre-requisitos**