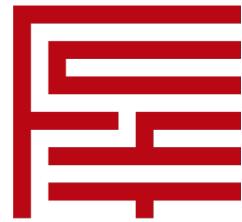


UNIVERSIDAD MAYOR DE SAN SIMÓN
FACULTAD DE CIENCIA Y TECNOLOGÍA
INGENIERIA INFORMÁTICA



**SISTEMA DE VIDEO VIGILANCIA INTELIGENTE PARA LA ALERTA INMEDIATA
ANTE SITUACIONES DE PELIGRO EN EL HOGAR**

Proyecto de Grado Presentado para optar el grado en Ingeniera Informática

Presentado por: Sergio Rodrigo Cárdenas Rivera

Tutor: Jorge Orellana Araoz

COCHABAMBA - BOLIVIA
Diciembre - 2022

Dedicatoria

Dedico este proyecto especialmente a mis padres porque sin su apoyo incondicional no lo hubiera logrado. Su compañía y bendición a lo largo de mi vida me reconforta y me lleva por el camino del bien. Por eso les entrego este trabajo en ofrenda por su paciencia y amor infinito. Los quiero mucho.-

Agradecimientos

Agradezco a la prestigiosa Universidad Mayor de San Simón y a su prestigiosa Facultad de Ciencia y Tecnología, por darme una oportunidad más para poner en práctica lo que he aprendido y desarrollar herramientas útiles para la sociedad. A mi familia, por todo el impulso y apoyo incondicional que me dieron para superar todos los obstáculos que se presentaron en mi vida hasta ahora.

Índice general

| | |
|---|------------|
| Dedicatoria | I |
| Agradecimientos | III |
| 1. Introducción | 1 |
| 1.1. Antecedentes | 2 |
| 1.2. Descripción del Problema | 2 |
| 1.2.1. Definición del problema | 2 |
| 1.3. Objetivos del Proyecto | 2 |
| 1.3.1. Objetivo General | 2 |
| 1.3.2. Objetivos Específicos | 2 |
| 1.4. Justificación | 3 |
| 1.5. Alcances y límites | 3 |
| 2. Marco Teórico | 5 |
| 2.1. Sistema de video vigilancia | 5 |
| 2.2. Arquitectura de red | 6 |
| 2.2.1. Protocolos | 8 |
| 2.2.2. Modelo cliente-servidor | 9 |
| 2.2.3. HTTP | 12 |
| 2.3. Inteligencia Artificial | 13 |
| 2.3.1. Redes Neuronales | 13 |
| 2.4. Machine Learning (Aprendizaje de Máquina) | 16 |
| 2.4.1. Tipos de Machine Learning | 16 |
| 2.5. Deep Learning (Aprendizaje profundo) | 17 |
| 2.6. Técnicas de visión por computadora | 17 |
| 2.6.1. Aplicaciones | 17 |
| 2.6.2. Librerías | 18 |
| 2.7. Streamming | 19 |
| 2.7.1. Servidor web para el Streaming de contenido multimedia | 19 |
| 2.8. Lenguaje de Programacion | 23 |
| 2.9. Metodología de desarrollo | 23 |
| 2.9.1. Modelo Cascada o Waterfall | 23 |
| 2.9.2. Ventajas y desventajas del modelo Cascada | 27 |

| | |
|--|-----------|
| 3. Seguridad en el hogar | 29 |
| 3.1. Introducción | 29 |
| 3.2. Ausencia en el hogar | 29 |
| 3.2.1. Ausencias cotidianas | 30 |
| 3.2.2. Ausencias de termino medio | 30 |
| 3.2.3. Ausencias prolongadas | 30 |
| 3.3. Situaciones de riesgo | 31 |
| 3.3.1. Presencia de intrusos | 31 |
| 3.3.2. Fuego y humo | 31 |
| 3.4. Sistemas de seguridad | 32 |
| 3.4.1. Alarmas | 32 |
| 3.4.2. Sensores | 33 |
| 3.4.3. Cámaras | 33 |
| 4. Análisis y Diseño | 35 |
| 4.1. Análisis | 35 |
| 4.1.1. Definición de Requerimientos | 36 |
| 4.2. Planificación | 37 |
| 4.3. Diseño | 38 |
| 4.3.1. Módulo de cámaras | 38 |
| 4.3.2. Módulo de servidor | 43 |
| 5. Implementación | 57 |
| 5.1. Módulo de Cámaras | 57 |
| 5.2. Módulo de servidor | 59 |
| 5.2.1. Interfaz de consola | 59 |
| 5.2.2. Configuración y conexiones | 59 |
| 5.2.3. Análisis de fotogramas | 60 |
| 5.2.4. Transmisión de video en vivo | 63 |
| 6. Pruebas | 65 |
| 6.1. Pruebas de conexión al servidor | 65 |
| 6.2. Pruebas de notificación | 65 |
| 6.3. Prueba de transmisión en vivo | 65 |
| 7. Conclusiones | 67 |
| 8. Ejemplos | 69 |
| 8.1. Mas ejemplos de formato | 69 |
| 8.1.1. Familia | 69 |
| 8.1.2. Forma | 69 |
| 8.1.3. Tamaño | 69 |
| 8.2. Listado | 69 |
| 8.2.1. No numerados | 69 |
| 8.2.2. Numerados | 70 |

| | |
|--|-----------|
| 8.3. Referenciación con APA | 70 |
| 8.3.1. Citación como parte de párrafo | 70 |
| 8.3.2. Citación en la parte final | 70 |
| 8.3.3. Citación con número de página | 70 |
| 8.3.4. Citación anexos | 70 |
| 8.4. Figuras | 71 |
| 8.5. Tablas | 71 |
| 8.5.1. Corto | 71 |
| 8.5.2. Multipágina | 71 |
| 8.6. Fórmulas matemáticas | 73 |
| 8.7. Diagramas de flujo | 74 |
| Referencias | 77 |
| Anexos | 80 |
| Anexo A: Manual de instalacion de la camara | 83 |
| Anexo B: Instalación del servidor | 85 |
| Anexo C: Instalación de la aplicación | 87 |

Índice de figuras

| | | |
|-------|--|----|
| 2.1. | Componentes de una sistema de video-vigilancia. | 5 |
| 2.2. | Proyección del mercado de la video-vigilancia. | 6 |
| 2.3. | Esquema de capas del modelo OSI. | 7 |
| 2.4. | Modelo TCP/IP frente al modelo OSI. | 8 |
| 2.5. | Ilustración del Modelo cliente-servidor. | 10 |
| 2.6. | Solicitud de conexión por medio de un socket. | 10 |
| 2.7. | Conexión establecida entre sockets. | 11 |
| 2.8. | Flujo de interacción entre servidor y cliente TCP. | 11 |
| 2.9. | Interacción del protocolo HTTP. | 12 |
| 2.10. | La Inteligencia Artificial y sus sub-áreas. | 13 |
| 2.11. | Ilustración de una Neurona Artificial. | 13 |
| 2.12. | Modelo de capas de una red neuronal. | 14 |
| 2.13. | Ilustración de una Red Neuronal Convolutacional. | 15 |
| 2.14. | Movimiento del kernel. | 15 |
| 2.15. | Diferencias entre programación clásica y M.L. | 16 |
| 2.16. | Logotipo de la librería. | 18 |
| 2.17. | Streaming en un servidor web. | 19 |
| 2.18. | Streaming en un servidor web. | 20 |
| 2.19. | Http Live Streaming. | 22 |
| 2.20. | Dinamyc Adaptative Streamming Over Http. | 22 |
| 2.21. | Logo del lenguaje de programación Python. | 23 |
| 2.22. | Diagrama del Modelo Cascada. | 24 |
| 2.23. | Modelo Cascada: Relación iterativa entre las fases sucesivas. | 25 |
| 3.1. | Ilustración ejemplo de un intruso. | 30 |
| 3.2. | Ilustración ejemplo de un ladrón. | 31 |
| 3.3. | Ilustración ejemplo de fuego en interiores. | 31 |
| 3.4. | Ilustración de la presencia de fuego y humo en una habitación cerrada. | 32 |
| 3.5. | Ilustración de alarmas con sonido. | 33 |
| 3.6. | Ilustración de detector de humo. | 33 |
| 3.7. | Ilustración de diversas cámaras de seguridad. | 33 |
| 3.8. | Cámara de vigilancia de interiores. | 34 |
| 4.1. | Diseño de interacción de los módulos del sistema de video-vigilancia. | 36 |
| 4.2. | Diagrama de Gannt. | 38 |

| | |
|--|----|
| 4.3. Diseño de interfaz gráfica de usuario (Módulo de cámaras). | 39 |
| 4.4. Diagrama de interacción entre el módulo de cámaras con el servidor. | 40 |
| 4.5. Diagrama de clases del módulo de cámaras. | 41 |
| 4.6. Diagrama de secuencia: Notificación de conexión/desconexión. | 44 |
| 4.7. Diagrama de secuencia: Detector de fuego. | 45 |
| 4.8. Diagrama de secuencia: Detector de silueta humana. | 46 |
| 4.9. Diagrama de secuencia: Detector de movimiento. | 47 |
| 4.10. Diagrama de secuencia: Streamming de video en vivo. | 48 |
| 4.11. Diagrama de clases: Servidor | 49 |
| 4.12. Diseño de notificación: Conecta una cámara más al sistema. | 53 |
| 4.13. Diseño de notificación: Desconexión de la única cámara. | 53 |
| 4.14. Diseño de notificación: Desconexión de una de las cámaras disponibles. | 53 |
| 4.15. Diseño de notificación: Nueva y única conexión. | 54 |
| 4.16. Diseño de notificación: Fuego detectado. | 54 |
| 4.17. Diseño de notificación: Silueta humana detectada. | 54 |
| 4.18. Diseño de notificación: Movimiento detectado. | 55 |
| 5.1. Interfaz gráfica del módulo de cámaras. | 57 |
| 5.2. Código de envío de un fotograma por medio de sockets. | 58 |
| 5.3. Interfaz gráfica del módulo de cámaras en envio de fotogramas. | 58 |
| 5.4. Ejecución del servidor TCP. | 59 |
| 5.5. Archivo de configuración del servidor. | 59 |
| 5.6. Mensajes de consola del servidor. | 60 |
| 5.7. Código de detección de fuego. | 61 |
| 5.8. Código de detección de silueta humana. | 62 |
| 5.9. Código de detección de movimiento. | 62 |
| 5.10. Código de configuración de la librería VidGear. | 63 |
| 8.1. Explicación de la figura (Aquí) | 71 |

Índice de tablas

| | |
|---|----|
| 4.1. Lista de requerimientos funcionales | 37 |
| 4.2. Tabla de planificación según fases del modelo Cascada. | 38 |
| 8.1. Título de la tabla | 71 |
| 8.2. Titulo de tabla multipágina | 72 |
| 8.3. Detalle de las pruebas realizadas | 75 |
| 8.4. Detalle de las pruebas realizadas | 75 |

Capítulo 1

Introducción

El término “Seguridad” es usado para hacer referencia a la ausencia de riesgo o a la confianza explícita en algo o alguien; pero este panorama toma diversos sentidos según el campo en el que se enfoca la seguridad. Aunque el objetivo consista en reducir el riesgo a niveles aceptables, el mismo es inherente a cualquier actividad o situación y en ninguna circunstancia puede ser eliminado.

Desde la aparición del hombre sobre la faz de la tierra, siempre prevaleció el instinto de supervivencia, donde surge la necesidad de obtener y/o brindar seguridad ante cualquier peligro que ponga en riesgo la integridad física propia y la de sus seres más cercanos. Cuando las primeras sociedades se formaron, una de las principales tareas del estado fue administrar justicia y brindar seguridad.

En el ámbito de la seguridad, la video-vigilancia se define como el acto de observar una escena o escenas en busca de comportamientos específicos que podrían ser anormales o podrían indicar una posible emergencia o existencia de un comportamiento impropio (Norman, 2017). Los sistemas de video-vigilancia en la actualidad, se han convertido en una herramienta esencial de la seguridad para mantener “observado/vigilado” un espacio muy importante para quien requiere el sistema; donde el mismo está compuesto por un conjunto de cámaras, monitores y grabadoras los cuales forman parte esencial del sistema. Estos sistemas pueden ser instalados tanto en interiores como en exteriores de una propiedad o establecimiento, especialmente en lugares donde se desea mantener una vigilancia constante.

Gracias a la tecnología actual se ha podido automatizar la mayoría de las tareas que los humanos realizan y el campo de la video-vigilancia no ha sido la excepción. Con los continuos avances tecnológicos cada vez se desarrollan sistemas más completos y avanzados, permitiendo incrementar su eficacia y confiabilidad; por ejemplo la capacidad de poder vigilar en la oscuridad gracias a la tecnología de visión nocturna. Pero el campo más fascinante dentro de estos avances es el de la Inteligencia Artificial y específicamente en la rama de la “Visión por Computadora”. Gracias a las técnicas utilizadas en este campo de investigación, una computadora con el apoyo de algoritmos específicos y clasificadores, posee la capacidad de identificar objetos, siluetas y/o elementos dentro de una escena captada por una cámara.

Estas características pueden ser aprovechadas en innumerables actividades diarias, donde es necesaria la supervisión de una persona, permitiendo aún más la automatización de tareas de vigilancia.

El problema principal de este escenario es evaluar si lo que esta siendo identificado en una escena representa un peligro para las personas.

1.1. Antecedentes

En la actualidad es común que empresas e instituciones tengan instalados sistemas de seguridad en sus ambientes como ser: oficinas, sitios de producción, almacenes, entradas, recepción, etc. pero realmente no solo las empresas tienen algún riesgo de situación de peligro o robo, si no también las personas en sus respectivos hogares.

Con el continuo crecimiento del mercado de la seguridad, el precio de los equipos de video-vigilancia tendieron a decrecer pero aun no son accesibles para todo el mundo. Este hecho asociado con el incremento de la inseguridad independientemente de cada país, promueve los siguientes escenarios: un incremento en el uso de sistemas de video-vigilancia, sistemas con varias cámaras funcionando al mismo tiempo siendo monitoreadas solo por un usuario el cual no esta disponible todo el tiempo y la ausencia de características avanzadas de reconocimiento de escenas en sistemas de video-vigilancia convencionales.

1.2. Descripción del Problema

Cuando el responsable del hogar esta ausente y nadie vigila, la posibilidad de acontecer una situación anormal siempre esta presente. Si en el peor de los casos llega a ocurrir algo en su hogar, esta persona solo se llega a enterar si algún vecino se comunica con él para avisarle lo sucedido o en el peor de los casos, enterarse directamente a su regreso. Un sistema de video-vigilancia con las características de identificar movimiento y situaciones de peligro como ser: presencia de intrusos y fuego, puede reducir el daño causado por los sucesos antes descritos por medio de la acción inmediata por parte del usuario en el momento de ser notificado, apoyado por la visualización en tiempo real de lo que estan captando las cámaras.

1.2.1. Definición del problema

Dificultad para advertir de forma inmediata situaciones de peligro en el hogar.

1.3. Objetivos del Proyecto

A continuación se presentan el objetivo general y los objetivos específicos.

1.3.1. Objetivo General

Facilitar la alerta inmediata ante situaciones de peligro en el hogar por medio de un sistema de video-vigilancia inteligente.

1.3.2. Objetivos Específicos

1. Describir todos los factores que implican el proceso de transmisión de datos por la red.

2. Especificar el proceso de análisis y procesamiento de imágenes con inteligencia artificial.
3. Proveer una red neuronal para el reconocimiento y análisis de video.
4. Identificar las partes que conforman el proceso de transmisión de video.
5. Describir medios para la interacción entre la transmisión y el análisis de imágenes.
6. Proveer el medio de acceso y notificación entre el sistema y el usuario.

1.4. Justificación

El riesgo de que un suceso ponga en peligro la integridad física y material de las personas esta presente cada día y en cualquier lugar. A pesar de que esta posibilidad es imposible de eliminar, se pueden crear mecanismos que contrarresten el impacto que ocasionan dichos sucesos en los sitios que se quieren evitar. Algunas situaciones más comunes que pueden representar un peligro a la integridad física y/o material del hogar son: la presencia de intrusos en ausencia del responsable en el hogar y la presencia de fuego y/o humo en el interior y/o exterior del hogar.

Los sistemas de video-vigilancia permiten la visualización en tiempo real de lo que las cámaras estan captando, pero es necesario una persona que ejecute la acción constante de revisar dicha transmisión para identificar y alertar sobre algunas situaciones que según su criterio pueden llegar a ser peligrosas. Si la cantidad de cámaras es considerable, la eficacia del operador del sistema disminuye al tener que revisar la transmisión de varias cámaras.

Con el aprovechamiento de la tecnología actual se plantea la implementación de un prototipo para un sistema de video-vigilancia inteligente que permita retransmitir de manera remota los fotogramas captados por las cámaras, con la característica de alertar al usuario sobre los sucesos antes descritos una vez que se identifican por medio de técnicas de visión por computadora y redes neuronales, para la acción inmediata del usuario con el fin de disminuir su impacto.

1.5. Alcances y límites

- La transmisión de fotogramas será implementado tanto para su ejecución en un ambiente local, como en línea.
- La notificación del evento identificado se realizará por medio de correo electrónico.
- La visualización en vivo del registro de la cámara se realizará por medio de un reproductor web de video con la capacidad de reproducir video en vivo HLS (HTTP Live Streaming).
- Se implementará la detección de: movimiento, silueta humana y fuego.
- Los fotogramas capturados serán analizados por medio de técnicas de visión artificial y redes neuronales.
- La transmisión de video en tiempo real se realizará por medio de un servidor web que implementa software libre en el streaming de video.

Capítulo 2

Marco Teórico

2.1. Sistema de video vigilancia

El término “video-vigilancia” es usado para hacer referencia al despliegue de cámaras de vídeo que cumplen el rol de videofilmadoras, las cuales guardan el contenido recolectado en un almacén digital y puede ser visualizado en un monitor central (Wikipedia, 2020). Por lo tanto, un sistema de video-vigilancia consiste en una instalación de seguridad cuya finalidad es el control y supervisión visual en tiempo real de instalaciones locales y remotas, mediante el uso de múltiples cámaras de vigilancia, así como de sistemas de visualización, grabación y archivo. Estos sistemas ayudan a proteger a las personas, bienes y recursos, manteniendo una alerta activa con un gran efecto disuasorio.

El sistema llega a capturar imágenes y videos, que pueden ser comprimidos, almacenados, o enviados por una red de comunicación y pueden ser instalados en cualquier ambiente. En la figura 2.1 se visualiza el conjunto de elementos que forman un sistema de video-vigilancia. Este sistema compone de un conjunto de cámaras que están conectadas directamente a un grabador de video en red o N.V.R. (Network Video Recorder), el cual permite la visualización de las imágenes captadas por las cámaras en un monitor local y por medio de una conexión a internet, permite su visualización en dispositivos externos a la red local.



Figura 2.1: Componentes de una sistema de video-vigilancia.
Fuente: (Solintel, 2018)

Existe una amplia oportunidad para el mercado de la video-vigilancia en todas las regiones del mundo especialmente en Asia y la región del Pacífico, debido a la apertura de pequeños negocios y la construcción de áreas residenciales como también ciudades “inteligentes” (MarketsAndMarkets,

2020). El mercado creciente de la vigilancia ha permitido que desarrolladores independientes y fabricantes diseñen nuevas implementaciones de sistemas de video-vigilancia, los cuales aplican nuevas características realizable gracias a la tecnología actual. En la figura 2.2 se muestra como el mercado global de la video-vigilancia tuvo un valor de 42.9 billones de dólares en el 2019 y esta proyectado alcanzar a los 69.1 billones de dólares hasta el 2026; cuyo incremento registra una taza de crecimiento anual compuesto del 10 % desde el 2020 al 2026. (MarketsAndMarkets, 2020)



Figura 2.2: Proyección del mercado de la video-vigilancia.

Fuente: (MarketsAndMarkets, 2020)

El aspecto más relevante del crecimiento en las oportunidades de este campo, se da en la implementación de nuevas características en este tipo de sistemas; gracias a las técnicas de visión por computadora e inteligencia artificial (I.A.), a la escalabilidad gracias al uso de servicios basados en la nube. Las ramas afines a la inteligencia artificial, como el Machine Learning (Aprendizaje Automático) y el Deep Learning (Aprendizaje profundo) han permitido lograr grandes avances en el campo de la video-vigilancia.

Para la implementación del prototipo propuesto, es necesario describir el marco teórico relevante en la captura de fotogramas de video, transmisión de datos por medio de la red por el protocolo tcp/ip, reconstrucción de información, consolidación y procesamiento de imágenes, como también la transmisión de video en vivo. A continuación se detallan los conceptos teóricos relevantes anteriormente descritos.

2.2. Arquitectura de red

Una arquitectura de red es un esquema completo de comunicación entre computadoras, el cual provee: un esquema de trabajo, un diseño principal, la construcción y manejo de una red. (Technologies Javvin, 2004, 1). La arquitectura de red más importante es la de Interconexión de Sistemas Abiertos (OSI), desarrollada por la Organización Internacional para la Estandarización (ISO).

La arquitectura OSI, es un estándar abierto para la comunicación en red a través de diferentes equipos y aplicaciones. Aunque no es ampliamente implementado; el modelo de 7 capas OSI es considerado el modelo de arquitectura de red principal para la intercomputación y comunicación entre redes. En la figura 2.3 se aprecia el modelo OSI de 7 capas y se detalla a continuación las siguientes capas:

1. Capa física (Physical)
2. Capa de enlace (Data Link)
3. Capa de red (Network)
4. Capa de transporte (Transport)
5. Capa de sesión (Session)
6. Capa de presentación (Presentation)
7. Capa de aplicación (Application)

Este modelo se organiza de la siguiente manera: las capas 7 a 4 se ocupan de las comunicaciones de extremo a extremo entre la fuente de datos y destinos, mientras que las capas 3 a 1 se ocupan de las comunicaciones entre los dispositivos de red. Por otro lado, las siete capas del modelo OSI pueden dividirse en dos grupos: **capas superiores** (capas 7, 6 y 5) y **capas inferiores** (capas 4, 3, 2, 1).

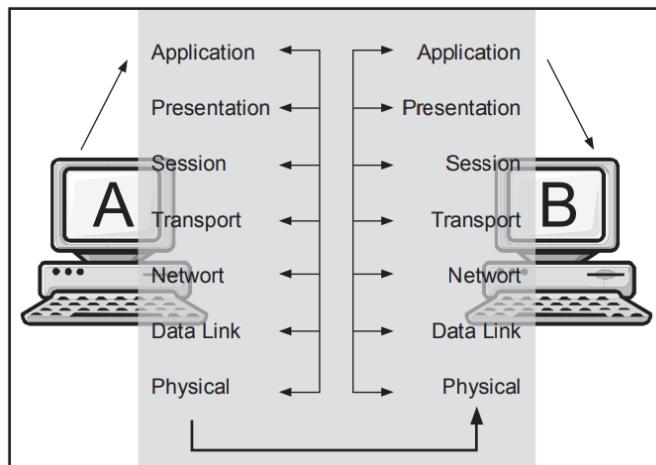


Figura 2.3: Esquema de capas del modelo OSI.
Fuente: (Technologies Javvin, 2004, 3)

Su contraparte en las arquitecturas de red del modelo OSI, es el modelo TCP/IP, que no sigue exactamente el modelo OSI. Desafortunadamente, no existe un acuerdo universal sobre cómo describir TCP/IP con un modelo en capas. Generalmente se acepta que TCP/IP tiene menos niveles (de tres a cinco capas) que las siete capas del modelo OSI. En la figura 2.4 se visualiza las capas

que se adoptan en esta arquitectura.

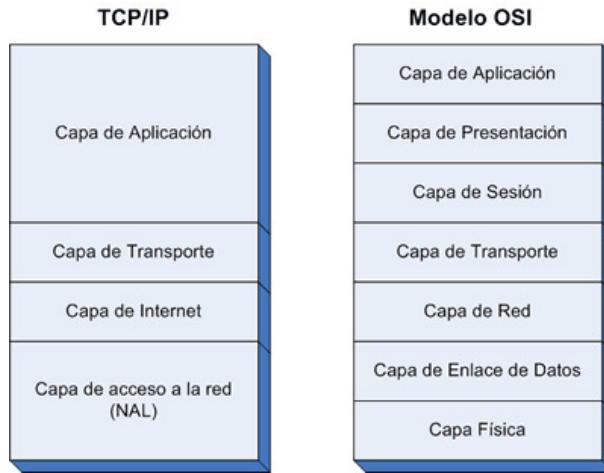


Figura 2.4: Modelo TCP/IP frente al modelo OSI.

Fuente: (Garza, 2013)

La arquitectura TCP/IP omite algunas características que se encuentran en el modelo OSI, combina las características de algunas capas OSI adyacentes y separa otras. La estructura de 4 capas de TCP/IP (capa de aplicación, transporte, internet y acceso a la red) se construye a medida que la información se transmite de la capa de aplicación a la capa de red física.

Cuando los datos se envían, cada capa trata la información que recibe de la capa superior como datos y agrega información de control (encabezado) al frente de esos datos y luego los pasa a la capa inferior. Cuando se reciben los datos, se lleva a cabo el procedimiento opuesto ya que cada capa procesa y elimina su encabezado antes de pasar los datos a la capa superior.

2.2.1. Protocolos

El modelo OSI, y cualquier otro modelo de comunicación de red, proporciona solo un esquema conceptual para la comunicación entre computadoras, pero el modelo en sí mismo no proporciona métodos específicos de comunicación (Wikipedia, 1994). La comunicación real está definida por varios protocolos de comunicación.

En el contexto de la comunicación de datos, un protocolo es un conjunto formal de reglas, convenciones y estructuras de datos que determinan cómo las computadoras y otros dispositivos de red intercambian información a través de una red. Este método estándar permite la comunicación entre procesos (que potencialmente se ejecutan en diferentes equipos) y agrega un conjunto de reglas y procedimientos que deben respetarse para el envío y la recepción de datos a través de una red.

Similar a la manera de hablar el mismo lenguaje entre dos personas; un protocolo, simplifica la comunicación entre dispositivos. La arquitectura de red proporciona solo un esquema conceptual para la comunicación. El modelo no proporciona métodos específicos de comunicación, sino mas

bien, la comunicación real está definida por varios protocolos de comunicación que son usados en la comunicación analógica y digital, y pueden ser usados en el procesos de transferencia de archivos y acceso a internet. Los protocolos de comunicación en red más populares, incluyen:

- **Automatización:** Automatizan diferentes procesos tanto en entornos comerciales como personales: en edificios inteligentes, tecnología en la nube o vehículos autónomos.
- **Mensajería Instantánea:** La comunicación basada en texto, en teléfonos inteligentes y computadoras suceden debido a una serie de diferentes protocolos de mensajería instantánea.
- **Enrutamiento:** Protocolos de enrutamiento permiten la comunicación entre routers y otros dispositivos de red.
- **Transferencia de archivos:** Envio de archivos por medio de un canal de comunicación.
- **Acceso a Internet:** El protocolo de Internet (IP) permite que los datos sean enviados entre dispositivos por medio de la red de internet.

A continuación se detallan algunos de los protocolos más conocidos:

- **HTTP - Protocolo de transferencia de hipertexto:** Este protocolo de internet define la manera en la que los datos son enviados por internet y determina como los navegadores web y buscadores deben responder a determinados comandos.
- **SSH - Secure Socket Shell:** Este protocolo provee un acceso seguro al dispositivo, incluso si se encuentra en una red no segura. SSH es particularmente usado por administradores de red quienes manejan diferentes sistemas de manera remota.
- **SMS - Servicio de envío de mensajes cortos:** Este protocolo ha sido creado para enviar y recibir mensajes de texto sobre redes de telefonía celular. SMS refiere exclusivamente a mensajes basados en texto. Las imágenes, videos u otro contenido multimedia requiere el protocolo de Servicio de mensajería multimedia (MMS), que es una extensión del protocolo SMS.
- **ICMP - Protocolo de control de mensajes de Internet:** Trabaja como asistente del protocolo de Internet y se encarga de identificar fallos en la información y enviar mensajes de error hacia el usuario o servidor. Por ejemplo, si una dirección de este no está disponible o si una solicitud presenta fallas.
- **SMTP - Protocolo de transferencia de correo simple:** Este se encarga del intercambio de datos por texto en mensajes de correo electrónico entre ordenadores por medio de la red.

2.2.2. Modelo cliente-servidor

El modelo cliente-servidor es una estructura de aplicación distribuida que divide tareas o cargas de trabajo entre los proveedores de un recurso o servicio, denominados servidores, y los solicitantes del servicio, denominados clientes (Wikipedia, 2022). A menudo, los clientes y los servidores se comunican a través de una red informática en hardware independiente, pero tanto el cliente como el servidor pueden residir en el mismo sistema.

Un servidor ejecuta uno o más programas de servidor, que comparten sus recursos con los clientes. Un cliente normalmente no comparte ninguno de sus recursos, pero solicita contenido o servicio de un servidor. En la figura 2.5 se aprecia un diagrama que representa el modelo cliente-servidor, donde los clientes acceden al servicio del servidor por medio de la misma red de internet.

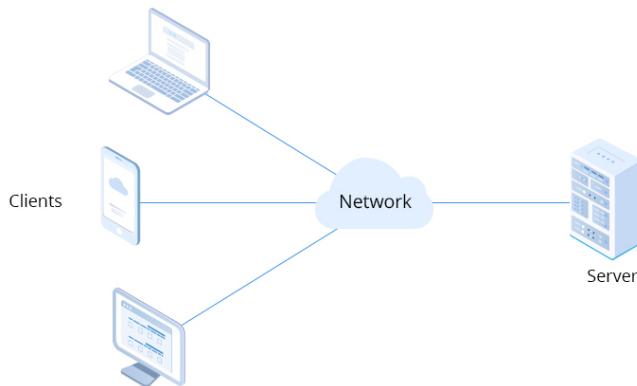


Figura 2.5: Ilustración del Modelo cliente-servidor.

Fuente: (Howard, 2022)

Por lo tanto los clientes, inician sesiones de comunicación con los servidores, que esperan las solicitudes entrantes. Para mencionar algunos ejemplos de aplicaciones informáticas que utilizan este modelo cliente-servidor son el correo electrónico, la impresión en red y la World Wide Web (Internet).

La comunicación entre el cliente y el servidor se realiza por medio de una conexión o enchufe “socket” el cual define la dirección y puerto por la cual va a ser enviada y/o recibida la información entre ambos actores. En la figura 2.6 se visualiza la forma en la que se realiza la conexión entre el lado del cliente y el servidor por medio de un conector o “socket”. En el lado del cliente: este conoce el nombre de host de la máquina en la que se ejecuta el servidor y el número de puerto en el que escucha el servidor.

Para realizar una solicitud de conexión, el cliente intenta conectarse con el servidor en la máquina y el puerto del servidor, además el cliente también necesita identificarse ante el servidor, para vincularse a un número de puerto local que se utiliza durante esa conexión. Normalmente lo asigna el sistema.



Figura 2.6: Solicitud de conexión por medio de un socket.

Fuente: (Oracle, 2015)

Si la solicitud tiene éxito, el servidor acepta la conexión. Después esta aceptación, el servidor obtiene un nuevo “socket” o conector vinculado al mismo puerto local como también tiene su punto final remoto establecido en la dirección y el puerto del cliente. En la figura 2.7 se visualiza el comportamiento del conector después de la aceptación a la petición de conexión del cliente; una vez establecida la conexión, el servidor necesita un nuevo socket para continuar escuchando las futuras solicitudes de conexión mientras atiende las necesidades del cliente conectado.



Figura 2.7: Conexión establecida entre sockets.

Fuente: (Oracle, 2015)

El flujo general de interacción es como sigue: el servidor TCP establece un conector en una determinada dirección y puerto, y se mantiene escuchando constantemente si algún conector cliente (que conoce la dirección y puerto del servidor) solicita una conexión; cuando el servidor acepta la conexión, se brinda un nuevo puerto diferente de manera automática para comunicarse exclusivamente con la nueva conexión y mantiene el puerto original a la escucha de nuevas conexiones, como se ve en la figura 2.8. Cuando se establece esta conexión, tanto como el cliente y el servidor pueden enviar y recibir datos. Cuando el cliente decide cerrar la conexión, el servidor pone de nuevo el puerto utilizado anteriormente como disponible.

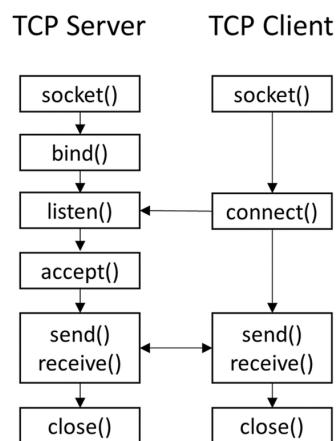


Figura 2.8: Flujo de interacción entre servidor y cliente TCP.

Fuente: (Gama et al., 2021)

A continuación se detalla los métodos que se muestran en la figura 2.8:

- `socket()`: Crea un punto final para la comunicación en el servidor.

- bind(): Asigna un número único al socket y reservar una combinación única de dirección IP y puerto para el socket creado.
- listen(): Espera a que un cliente se conecte.
- accept(): Recibe una solicitud de conexión de un socket de cliente.
- connect(): El cliente y el servidor están conectados entre sí.
- send()/recieve(): Intercambian datos entre el cliente y el servidor
- close(): El servidor y el cliente cortan la conexión.

Los clientes y servidores intercambian mensajes en un patrón de mensajería de solicitud-respuesta. Para comunicarse, las computadoras deben tener un lenguaje común y deben seguir reglas para que tanto el cliente como el servidor sepan qué esperar. El idioma y las reglas de comunicación se definen en un protocolo de comunicaciones. Todos los protocolos operan en la capa de aplicación.

2.2.3. HTTP

El protocolo de transferencia de hipertexto (HTTP) es el más utilizado en Internet. Se usa en cada transacción de la Web (www) y permite la transferencia de archivos (principalmente, en formato HTML) entre un navegador (cliente) y un servidor web. HTTP define la sintaxis y la semántica que utilizan los elementos software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Este protocolo se orienta a las transacciones y sigue el esquema “Petición - Respuesta” entre un cliente y un servidor. En la figura 2.9 se aprecia la interacción entre el cliente y el servidor web.

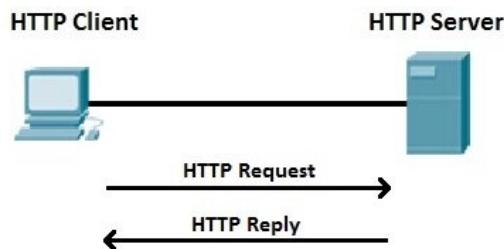


Figura 2.9: Interacción del protocolo HTTP.
Fuente: (CCNA-Certification, 2022)

Este protocolo se compone las siguientes características:

- El cliente que efectúa la petición (un navegador o un spider) es denominado “user agent” (agente del usuario).
- La información transmitida se denomina “recurso” y se identificada mediante una cadena de caracteres definida como dirección URL.
- Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc.

2.3. Inteligencia Artificial

La Inteligencia Artificial (I.A.), como área de estudio de las ciencias de la computación, en los últimos tiempos dejó de estar reservada para la investigación y pasó a formar parte del desarrollo de la sociedad. El cerebro es el órgano más increíble del cuerpo humano; establece la forma en la que percibimos las imágenes, sonido, olores, sabores y el tacto; por lo tanto permite al ser humano almacenar recuerdos, experimentar emociones e incluso soñar. Sin él, el ser humano sería un organismo primitivo, incapaz de otra cosa que el más simple de los reflejos. Por lo tanto, el cerebro es lo que hace inteligente al ser humano (Lopez, 2017).

En la construcción de estas máquinas artificialmente inteligentes, se presentan problemas computacionales complejos; problemas que el cerebro humano puede resolver en una fracción de segundos. Las formas de analizar y resolver este tipo de problemas, son estudiados por la inteligencia artificial. En la figura 2.10 se ilustra como la Inteligencia Artificial engloba los subcampos de estudio como son el Machine Learning y el Deep Learning.

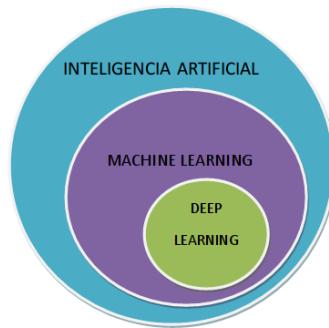


Figura 2.10: La Inteligencia Artificial y sus sub-áreas.

Fuente: (Tejada, 2019)

2.3.1. Redes Neuronales

Las redes neuronales artificiales, son un modelo inspirado en el funcionamiento del cerebro humano. Está formado por un conjunto de nodos conocidos como neuronas artificiales que están conectadas y transmiten señales entre sí. Estas señales se transmiten desde la entrada hasta generar una salida. En la figura 2.15 se aprecia la estructura propia de una neurona artificial.

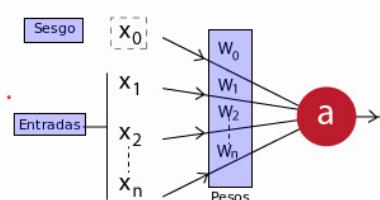


Figura 2.11: Ilustración de una Neurona Artificial.

Fuente: (Lopez, 2016)

El objetivo principal de un modelo neuronal, es aprender por medio de la modificación automática de si misma, llegando a realizar tareas complejas que no podrían ser realizadas mediante programación clásica basada en reglas. De esta forma se pueden automatizar funciones que al principio solo podrían ser realizadas por personas. Semejante al del cerebro humano, las redes neuronales reciben una serie de valores de entrada, donde cada una de estas llegan a un nodo llamado neurona.

Las neuronas de una red están a su vez agrupadas en capas formando una red neuronal. Cada una de las neuronas de la red poseen un peso, que es un valor numérico con el que modifica la entrada recibida. Los nuevos valores obtenidos por las neuronas continúan su camino por la red. En la figura 2.12 se aprecia el funcionamiento anteriormente descrito.

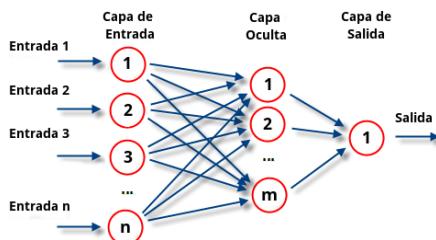


Figura 2.12: Modelo de capas de una red neuronal.

Fuente: (Wikipedia, 2012)

Cuando este proceso llega al final de la red, se obtiene una salida que será la predicción calculada por la red. Cuantas más capas posea la red y aumente la complejidad de sus conexiones; mas complejas serán las funciones que pueda realizar. Para que una red neuronal realice las funciones deseadas, es necesario la ejecución del entrenamiento correspondiente. El entrenamiento de la red neuronal se realiza por medio de la modificación de los pesos de sus neuronas para que consiga extraer los resultados deseados. Para ello, se introduce datos de entrenamiento en la red, según el resultado obtenido y el error se modifican los pesos de las neuronas en función de cuanto haya contribuido cada neurona a dicho resultado. Este método es conocido como “backpropagation” o propagación hacia atrás. Con este método se consigue que la red “aprenda”, consiguiendo un modelo capaz de obtener resultados muy acertados incluso con datos muy diferentes a los de su entrenamiento (Innovation, 2019).

Estas redes neuronales son utilizadas para tareas de predicción y clasificación. Esta técnica se ha convertido en una pieza clave para el desarrollo de la Inteligencia Artificial. Como se describió previamente, es uno de los principales campos de investigación y el que más ha evolucionado con el tiempo, ofreciendo cada vez soluciones más complejas y eficientes.

Redes neuronales convolucionales

Dentro del conjunto de tipos de redes neuronales están las convolucionales, que se usan en el campo de la visión artificial. Las redes neuronales convolucionales representan un algoritmo de Aprendizaje Profundo (Deep Learning) que está diseñado para trabajar con imágenes. Toma las imágenes como entrada, les asigna un valor (peso) a ciertos elementos en la imagen para su clasificación.

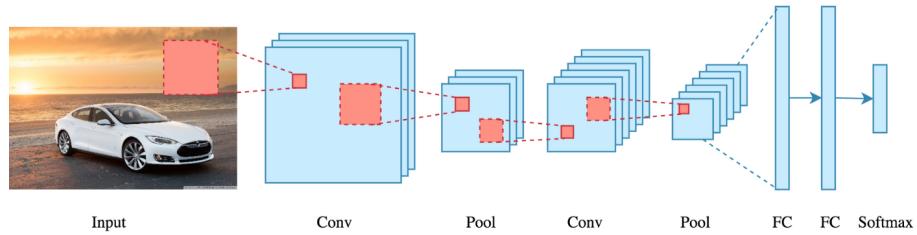


Figura 2.13: Ilustración de una Red Neuronal Convolucional.

Fuente: (Bootcamp-ia, 2019b)

Estas redes han contribuido en el desarrollo y perfeccionamiento del campo de la visión por computadora. Las redes convolucionales contienen varias capas ocultas como se ilustra en la figura 2.12, donde las primeras puedan detectar líneas, curvas y así se van especializando hasta poder reconocer formas complejas como un rostro, siluetas, etc. (Saha, 2018).

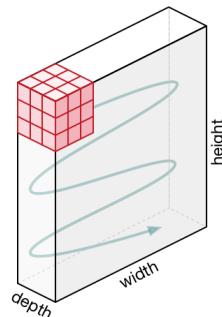


Figura 2.14: Movimiento del kernel.

Fuente: (Bootcamp-ia, 2019a)

Estas redes funcionan en base de convoluciones. Este proceso consiste en tomar un grupo de píxeles de la imagen de entrada e ir realizando un producto escalar con un kernel. El kernel recorrerá todas las neuronas de entrada y obtendremos una nueva matriz, la cual será una de las capas ocultas. En el caso de que la imagen sea de color se tendrán 3 kernels del mismo tamaño que se sumarán para obtener una imagen de salida.

El “kernel” dentro las redes convolucionales es considerado como un filtro que se aplica para extraer ciertas características importantes o patrones. Se usa para detectar bordes, enfoque, desenfoques, y características de la imagen por la convolución entre la imagen y el kernel. Este proceso se desarrolla entre la imagen y el kernel, con la finalidad de que el kernel recorra toda la imagen (pixel por pixel). Por lo general, el kernel es de menor tamaño que la imagen permitiendo multiplicar el kernel con la porción de imagen escogida, realiza un producto escalar a medida que el kernel se va desplazando; por esta razón es un proceso iterativo. Las tareas comunes de este tipo de redes son: detección o categorización de objetos, clasificación de escenas y clasificación de imágenes en general.

2.4. Machine Learning (Aprendizaje de Máquina)

Es un subcampo de la Inteligencia Artificial cuyo objetivo es entender la estructura de la información y ajustar estos datos en modelos que puedan ser entendidos y utilizados por las personas. (Tagliaferri, 2017). A diferencia de la computación tradicional, donde los algoritmos resuelven problemas específicos, los algoritmos de Machine Learning entran a las computadoras con datos de entrada y emplean análisis estadístico para generar valores de salida que se clasifican según a un rango específico. Por eso el Machine Learning facilita a las computadoras construir modelos a partir de datos ejemplo para automatizar el proceso de toma de decisiones basados en estos datos.

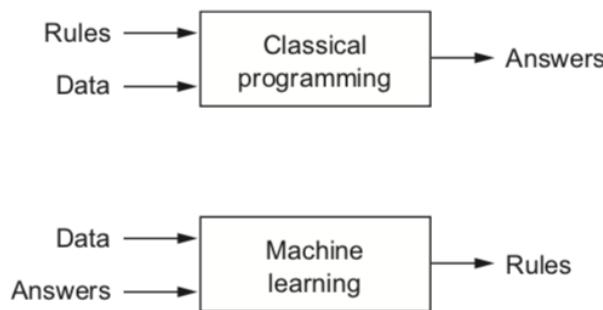


Figura 2.15: Diferencias entre programación clásica y M.L.

Fuente: (Chollet y Allaire, 2018)

En la figura 2.15 se visualiza la diferencia y similitud entre la programación clásica de la inteligencia artificial de los años '80 y las novedosas técnicas del aprendizaje automático. La programación clásica necesita de reglas y datos de entrada para que esta funcione como un sistema inteligente y pueda dar una respuesta, mientras que el Machine Learning necesita de datos y sus respectivas respuestas esperadas, para identificar patrones que los relacionen y así desarrollar reglas que generan respuestas para nuevos datos.

2.4.1. Tipos de Machine Learning

En el Machine Learning, las tareas se clasifican en grandes categorías, las cuales están basadas en el modo en el que el “aprendizaje” es ejecutado. Los métodos más adoptados en el Machine Learning son: el aprendizaje supervisado y el no supervisado.

Aprendizaje Supervisado

La computadora es provista de entradas ejemplo las cuales se categorizan con sus respectivas salidas esperadas. El propósito de este método consiste en que el algoritmo pueda “aprender” comparando la actual salida con las salidas esperadas para encontrar errores y en consecuencia modificar el modelo. El aprendizaje supervisado por lo tanto usa patrones para predecir valores categorizados en datos no categorizados.

Aprendizaje No Supervisado

La información provista a la computadora no está categorizada, por lo que los algoritmos de aprendizaje buscan similitudes entre los datos de entrada. Como los datos no etiquetados son más abundantes que los datos etiquetados, los métodos de aprendizaje automático que facilitan el “aprendizaje” pasan a ser más importantes.

2.5. Deep Learning (Aprendizaje profundo)

En la figura 2.10, se visualiza que el aprendizaje profundo es un subcampo dentro del Machine Learning, el cual hace uso de distintas redes neuronales para lograr el “aprendizaje” de sucesivas capas de representación que son relevantes para los datos. El término Deep “profundo”, hace referencia a la cantidad de capas de representación que se utilizan en un modelo; en general es posible utilizar decenas incluso cientos capas de representación, los cuales aprenden de forma automática a medida que el modelo es entrenado con los datos (Briega, 2015).

2.6. Técnicas de visión por computadora

La visión por computadora es una técnica de recolección de información que surge por la inspiración en el sistema visual humano, el cual es la principal fuente de información para el cerebro. Su meta es de modelar y automatizar el proceso de reconocimiento visual de objetos en la vida real.

De los cinco sentidos que poseen las personas, la vista es la más importante. Por lo tanto la visión, es una tarea de procesamiento de información; pero tiene un grado de complejidad elevado, saber que es lo qué hay en el mundo, el cerebro humano deben ser capaz de representar esta información sobre el color, la forma, movimiento, detalle y belleza de los objetos. (Briega, 2015)

La visión por computadora o visión artificial compone de un conjunto de herramientas y métodos que permiten obtener, procesar y analizar imágenes del mundo real por medio de una computadora. Estos métodos van a permitir automatizar un amplio conjunto de tareas al aportar a las computadoras información que es necesaria para la toma de decisiones en sus tareas asignadas. Esta técnica trata de imitar a la visión humana, usando geometría y un enfoque estadístico para solucionar el problema.

2.6.1. Aplicaciones

Esta rama de la Inteligencia Artificial aún sigue en investigación y mejoras donde sus aplicaciones más comunes son:

- **Reconocimiento óptico de caracteres:** Detección automática de símbolos que pertenecen a un alfabeto.
- **Inspección robotizada:** Revisión rápida de piezas para garantizar la calidad de componentes fabricados.
- **Modelado 3D:** Construcción de modelos 3D a partir de fotografías.
- **Imágenes médicas:** Análisis de radiografías.

- **Conducción segura:** Detección de obstáculos por medio de un sistema de conducción asistida por cámaras.
- **Vigilancia:** Monitoreo de intrusos, análisis del tráfico vial, monitoreo de piscinas, etc.
- **Detección de rostros:** Mediante algoritmos de reconocimiento facial se reconocen rostros usados en métodos de biometría.

2.6.2. Librerías

Una de las librerías mas utilizadas para las técnicas de visión por computadora es OpenCV. Es una biblioteca de uso libre para el desarrollo de aplicaciones usando visión artificial desarrollada por Intel. Esta librería reune diversas características que la hacen popular, por ejemplo:

- Permite su uso para fines comerciales y de investigación.
- Se encuentra disponible para varias plataformas como ser GNU/Linux, Mac OS, Windows y Android.
- Documentación completa y explicada, con una comunidad de desarrolladores activa.
- El procesamiento de imágenes en su escalado, eliminación de ruido y formateo de imagen y video.
- El uso y modificación de sus 2500 modelos pre-optimizados que son incluidos en la librería, acorde a las necesidades del usuario.
- El uso del estado del arte de modelos de visión por computadora como también de aprendizaje de máquina (Machine Learning).
- El desarrollo de modelos en varias categorías de investigación como ser: reconocimiento facial, detección y seguimiento de objetos, extracción de modelos 3D, etc.



Figura 2.16: Logotipo de la librería.
Fuente: Web

Existen otras librerías que no son tan populares pero representan un pago adicional.

2.7. Streaming

Este término también se denomina como transmisión, lectura en continuo, difusión en flujo, lectura en tránsito, difusión en continuo, descarga continua. Es la distribución digital de multimedia a través de una red de computadoras, de manera que el usuario consume el producto (generalmente archivo de vídeo o audio) en paralelo mientras se descarga (Austerberry, 2005).

El streaming requiere una conexión por lo menos de igual ancho de banda que la tasa de transmisión del servicio. El streaming de vídeo se popularizó a fines de la década de 2000, cuando el ancho de banda se hizo lo suficientemente barato para gran parte de la población. Sin embargo, con la tecnología del streaming, un archivo puede descargarse y reproducirse al mismo tiempo, con lo que el tiempo de espera es mínimo (Austerberry, 2005). El streaming permite la reproducción directa del contenido multimedia, sin la necesidad de almacenar en memoria la información para después reproducirla, como se puede ver en la figura 2.17.

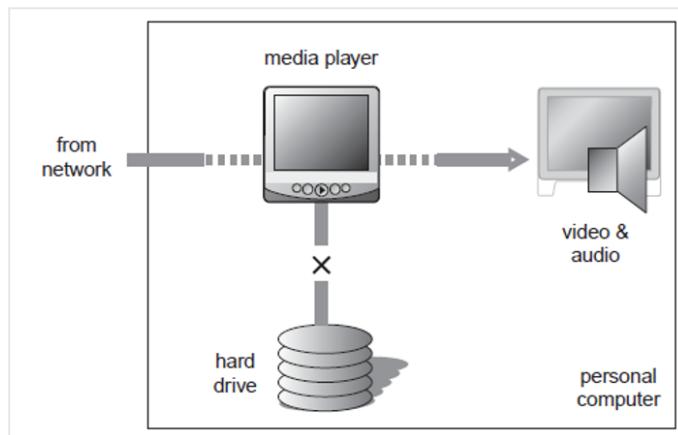


Figura 2.17: Streaming en un servidor web.

Fuente: (Austerberry, 2005)

El servidor más utilizado para la transmisión de contenidos multimedia es el servidor web, tipificado por Apache. Los servidores Web utilizan HTTP a través de TCP/IP para entregar sus páginas HTML y archivos de imagen asociados. El protocolo TCP/IP se utiliza como la capa de transporte a través de Internet. Los archivos se descargan a la memoria caché del navegador web más rápido de lo que el sistema permite. TCP incorpora control de flujo para gestionar la velocidad de descarga.

2.7.1. Servidor web para el Streaming de contenido multimedia

El servidor web reproduce contenido multimedia de la siguiente forma: el servidor mediante una aplicación proporciona una dirección que permite acceder a un archivo multimedia por medio de un reproductor. Luego que se establece una conexión al servidor web mediante el mismo enlace, se manda una señal para que el servidor transmita el contenido. El servidor web redirige al reproductor a un servidor streaming para luego recién realizar la reproducción del contenido digital, como se observa en la figura 2.18.

Un servidor web no realiza este proceso sin un servidor Streaming de por medio; permite la descarga del archivo en su totalidad para que después se realice la reproducción por parte del reproductor. El servidor streaming decodifica el bitrate en el que se codificó. Los oyentes deben tener instalado un reproductor multimedia en su ordenador, pero también se puede escuchar la emisión sin necesidad de éste último. Se podría utilizar: Windows Media Player, VLC, o algún software compatible.

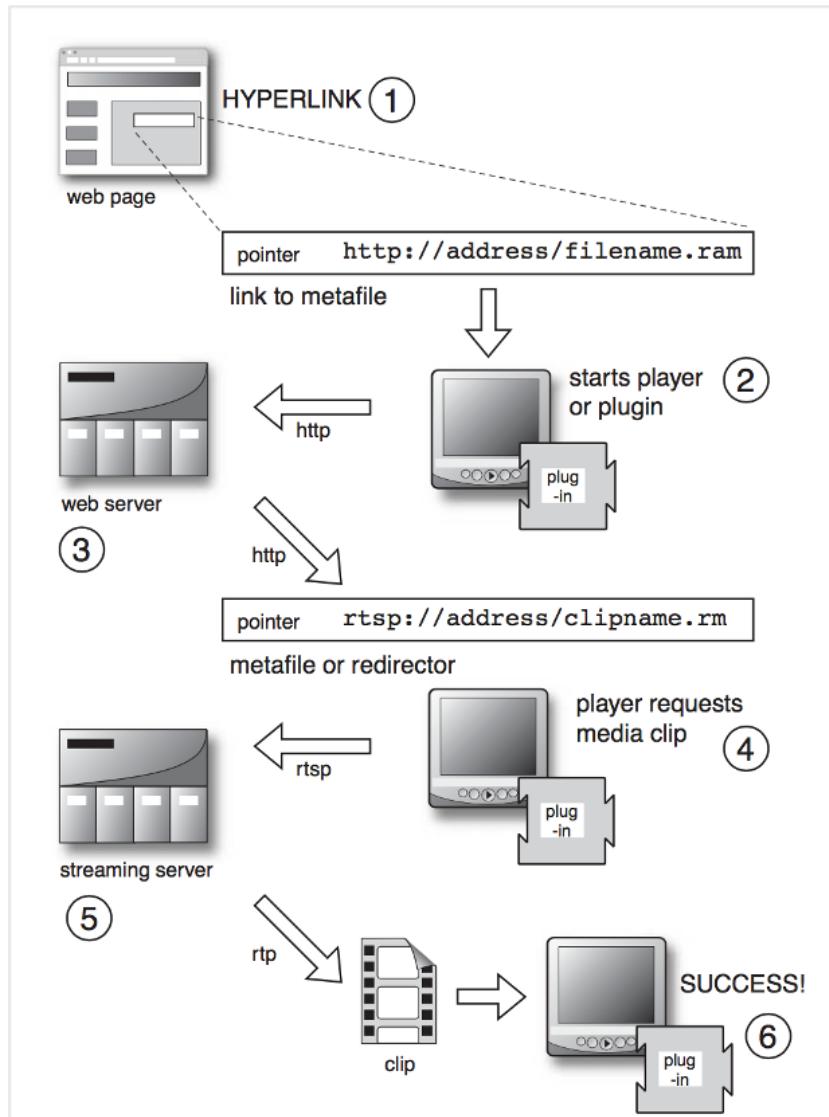


Figura 2.18: Streaming en un servidor web.

Fuente: (Austerberry, 2005)

El término bitrate se refiere a la calidad de la señal, usualmente se utilizan 24, 32, 48, 64 kbps pero si se requiere se puede emitir a un bitrate superior, por ejemplo 96 kbps o 128 kbps. Se debe tener en cuenta que a mayor bitrate es necesario mayor ancho de banda, así como mayor será el costo. Sin la tecnología de Streaming, para reproducir un archivo multimedia, sería necesario descargar en memoria del computador para después recién reproducirlo.

Http Live Streaming (HLS)

HLS es un protocolo utilizado para enviar audio y video sobre HTTP, soporta bitrate adaptivo y es compatible con la mayoría de los dispositivos. Este protocolo fue desarrollado por Apple y puede ser utilizado tanto para video en vivo como video en demanda. Entre sus características importantes se tiene:

- Se adapta a las condiciones de la red. Según el ancho de banda de internet del espectador, entrega una calidad o bitrate adecuado.
- Puede ser entregado desde un servidor tradicional de HTTP.
- Es protegido con encriptado o autenticación con el fin de proteger derechos de autor.

El flujo del protocolo es como sigue:

1. El archivo de video se codifica con compresión de video H.264 y compresión de Audio AAC.
2. Selecciona MPEG2-TS como contenedor de video, esto lo hace normalmente un encoder de video.
3. Cuando es codificado el video, se realiza un proceso de segmentación. Este proceso realiza cortes del video cada ciertos segundos, usualmente cada 10 segundos. A estos fragmentos se les conoce como segmentos o "chunks" de video.
4. Se genera un archivo índice. Este archivo contiene la información de los segmentos y en donde se encuentran para que el player los descargue. Este mismo proceso se realiza tanto para video en vivo o video en demanda (video almacenado).
5. Cuando se decide encriptar la información, se ejecuta un algoritmo para proteger cada uno de los segmentos. Donde el player que tenga la llave para desencriptarlos puede ver el contenido.
6. Estas llaves pueden ser estáticas o dinámicas. Puede generarse una llave para todos los videos o una llave por video.

La URL de reproducción tiene la siguiente forma:

`http://nombre-del-dominio/directorio-del-video/mi-video.m3u8`

Usando el mismo protocolo que impulsa la web, HLS comparte contenido usando servidores web y redes de entrega de contenido ordinarios. HLS está diseñado para brindar confiabilidad y se adapta dinámicamente a las condiciones de la red al optimizar la reproducción para la velocidad disponible de las conexiones por cable e inalámbricas.

En la figura, 2.19 se visualiza un esquema que representa el proceso de que sigue HLS. El archivo índice ".m3u8" permite a los reproductores de video descargar partes del contenido a medida que se reproduce, según el ancho de banda disponible en ese momento. Si el ancho de banda es bajo se realizan peticiones del video de una calidad más baja para no interrumpir la reproducción, caso contrario se realizan peticiones a la calidad normal o más alta.

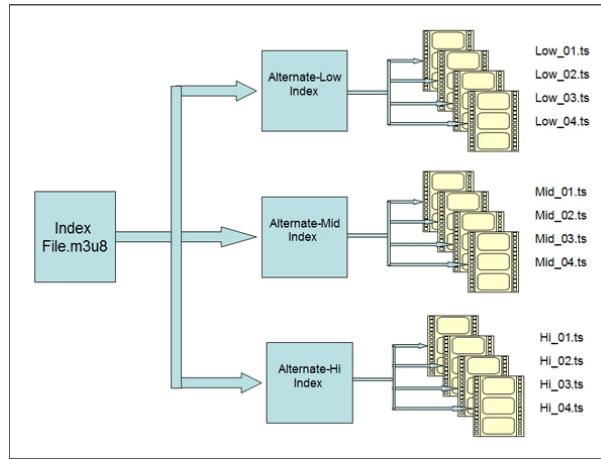


Figura 2.19: Http Live Streaming.

Fuente: (Ozer, 2017)

DASH

MPEG DASH (Dynamic Adaptive Streaming over HTTP) es un estándar para la transmisión adaptativa a través de HTTP que tiene el potencial de reemplazar tecnologías patentadas existentes como Microsoft Smooth Streaming, Adobe Dynamic Streaming y Apple HTTP Live Streaming (HLS) (Ozer, 2018). Todas las tecnologías de transmisión adaptativa basadas en HTTP utilizan una combinación de archivos multimedia codificados y archivos de manifiesto que identifican transmisiones alternativas y sus respectivas URL.

En la figura 2.20 se visualiza el funcionamiento del estándar DASH. El reproductor realiza las peticiones necesarias a un enlace el cual esta disponible y gestiona las peticiones de los clientes como del servidor en el momento de guardar los videos codificados.

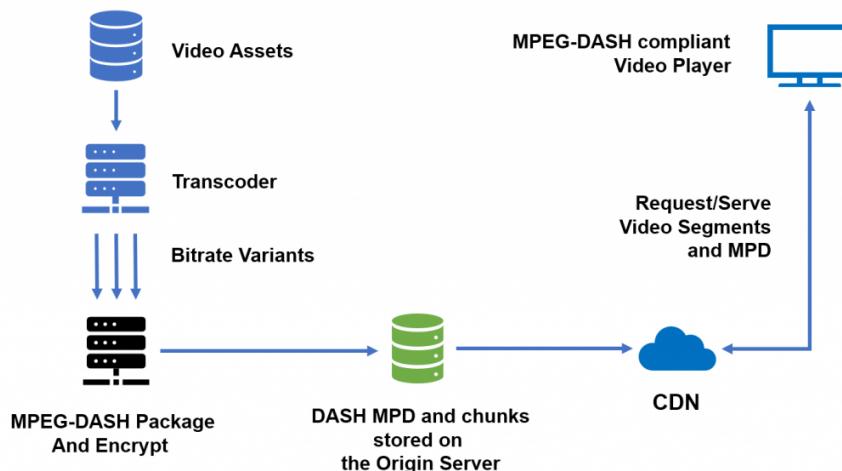


Figura 2.20: Dinamyc Adaptative Streamming Over Http.

Fuente: (Vijayanagar, 2021)

2.8. Lenguaje de Programación

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.



Figura 2.21: Logo del lenguaje de programación Python.

Fuente: Web

Python usa tipado dinámico y conteo de referencias para la administración de memoria. Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos). Actualmente es el lenguaje más popular y de amplio alcance en la inteligencia artificial (Henry, 2022).

2.9. Metodología de desarrollo

El concepto de ingeniería de software es propuesto por primera vez en el conjunto de conferencias históricas organizadas por el comité de ciencia de la OTAN.¹ en los años 60. Para ese tiempo la ingeniería de software tampoco era conocida ni aceptada. En ese entonces los proyectos de software complejos eran problemáticos y costosos de completar, entonces se supuso que sería beneficioso considerar el desarrollo de software como Ingeniería (Ganis, 2010).

Los encargados de la codificación del software llamados programadores; en un principio eran ingenieros, pero como el costo computacional era alto, se utilizó un concepto de hardware denominado "mide dos veces, corta una vez" (Ganis, 2010). La naturaleza cautelosa de esta costumbre provocó el desarrollo de metodologías que permitieron a los equipos de proyectos creen planes lentos y metódicos para la creación de sistemas de software.

2.9.1. Modelo Cascada o Waterfall

En sus inicios, este concepto fue abordado por el Dr. Winston Royce, por medio de un artículo escrito sobre la gestión y dirección de proyectos grandes y complejos de software (Royce, 1970). En ese artículo en base en experiencias de desarrollo de Software para la planificación de misiones aereas; el autor describe los fundamentos del desarrollo de Software. Gran parte de esos fundamentos aún son aplicables en la actualidad. En el planteamiento de estos fundamentos, Royce presenta un conjunto de fases los cuales forman parte de una secuencia de desarrollo de software ilustrada en la figura 2.22.

¹Organización del Tratado del Atlántico Norte.

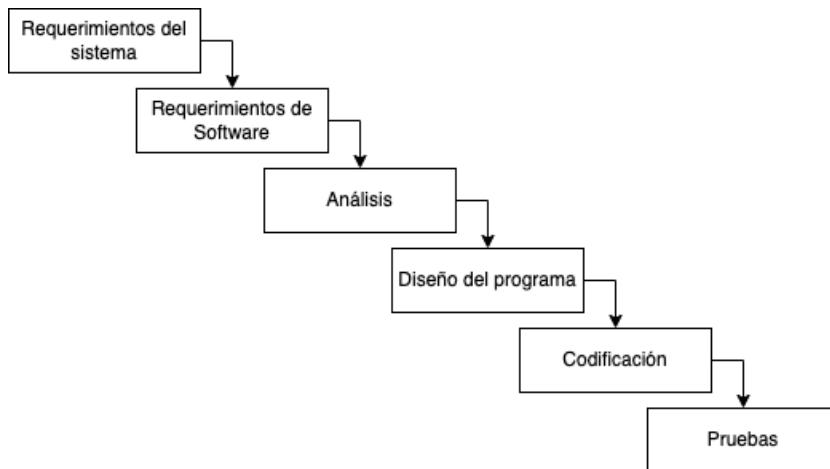


Figura 2.22: Diagrama del Modelo Cascada.

Fuente: Elaboracion Propia

Una vez definida esta secuencia, se crea el concepto de “Cascada”, como un modelo de desarrollo con actividades bien definidas y organizadas con un objetivo independiente dando origen al diseño del primer S.D.M.² (Bell y Thayer, 1976). En la figura 2.22 la fase de análisis y la de codificación entregan la mayor parte del producto esperado, mientras que las otras fases están puestas para ser organizadas y planificadas de manera independiente para un mejor manejo de los recursos del proyecto.

De acuerdo al modelo Cascada, se enfatiza en la dependencia secuencial del producto entregado en el paso previo. Es decir existe una dependencia que mantiene en espera el diseño del sistema mientras que el análisis del modelo no sea aprobado o concluido y consecuentemente la fase de codificación se verá en espera también hasta que el diseño se concluya.

Cada una de las fases guarda una relación iterativa con el siguiente paso definido en la metodología que asegura la completitud del producto entregado en la fase anterior. Esta relación está ilustrada en la figura 2.23. Al final de cada etapa, el modelo está diseñado para llevar a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase. Este modelo fue el primero en originarse y es la base de todos los demás modelos de ciclo de vida dentro de un proyecto de desarrollo de software.

En la práctica, se aplican diversas versiones del modelo. Los más habituales son los modelos que dividen los procesos de desarrollo en cinco fases. En ocasiones, las fases 1, 2 y 3 definidas por Royce se integran en una sola fase de proyecto a modo de análisis de los requisitos.

1. Análisis: planificación, análisis y especificación de los requisitos.
2. Diseño: diseño y especificación del sistema.

²Metodología de Desarrollo de Software

3. Implementación: programación y pruebas unitarias.
4. Verificación: integración de sistemas, pruebas de sistema y de integración.
5. Mantenimiento: entrega, mantenimiento y mejora.

La figura 2.23 se muestra la ampliación de esta metodología donde se añaden funciones iterativas al modelo básico como, por ejemplo, los saltos hacia atrás, que permiten comparar los resultados de cada una de las fases con las hipótesis obtenidas en la fase anterior, de modo que se puedan verificar.



Figura 2.23: Modelo Cascada: Relación iterativa entre las fases sucesivas.

Fuente: (Mancuzo, 2021)

En este modelo, las diferentes fases de un proceso de desarrollo se suceden una detrás de otra como en una cascada. Cada una de las fases concluye con un resultado provisional o “hito”. A continuación se detalla cada una de las fases y las actividades que implica cada fase.

Análisis

Todo proyecto de software comienza con una fase de análisis que incluye un estudio de viabilidad y una definición de los requisitos. En el estudio de viabilidad se evalúan los costes, la rentabilidad y la factibilidad del proyecto de software. El estudio de viabilidad da como resultado un pliego de condiciones (una descripción general de los requisitos), un plan y una estimación financiera del proyecto, así como una propuesta para el cliente, si fuera necesario.

Este es el punto de partida en donde se plasma cada detalle de la idea y tomar el impulso necesario para su desarrollo. Esta fase consiste en determinar cuáles son las necesidades y los objetivos, para luego reunir todos los requisitos que se deben cumplir en el desarrollo del software para llevar a cabo todo el proceso.

Se realiza una definición detallada de los requisitos, incluyendo un análisis de la situación de salida y un concepto. Mientras que los análisis de salida se encargan de describir la problemática en sí, el concepto ha de definir qué funciones y características debe ofrecer el producto de software para cumplir con las correspondientes exigencias. La definición de los requisitos da como resultado un pliego de condiciones, una descripción detallada de cómo se han de cumplir los requisitos del proyecto, así como un plan para la prueba de aceptación, entre otros.

Por último, la primera fase del modelo cascada incluye un análisis de la definición de los requisitos en el que los problemas complejos se dividen en pequeñas tareas secundarias y se elaboran las correspondientes estrategias de resolución.

Diseño

La fase de diseño sirve para formular una solución específica en base a las exigencias, tareas y estrategias definidas en la fase anterior. En esta fase, los desarrolladores se encargan de diseñar la arquitectura de software, así como un plan de diseño detallado del mismo, centrándose en componentes concretos, como interfaces, entornos de trabajo o bibliotecas. La fase de diseño da como resultado un borrador preliminar con el plan de diseño del software, así como planes de prueba para los diferentes componentes.

Implementación

La arquitectura de software concebida en la fase de diseño se ejecuta en la fase de implementación, en la que se incluye la programación del software, la búsqueda de errores y las pruebas unitarias. En la fase de implementación, el proyecto de software se traduce al correspondiente lenguaje de programación. Los diversos componentes se desarrollan por separado, se comprueban a través de las pruebas unitarias y se integran poco a poco en el producto final. La fase de implementación da como resultado un producto de software que se comprueba por primera vez como producto final en la siguiente fase (prueba alfa).

Verificación

La fase de verificación o pruebas incluye la integración del software en el entorno seleccionado. Por norma general, los productos de software se envían en primer lugar a los usuarios finales seleccionados en versión beta (pruebas beta). Las pruebas de aceptación desarrolladas en la fase de análisis permiten determinar si el software cumple con las exigencias definidas con anterioridad. Aquellos productos de software que superan con éxito las pruebas beta están listos para su lanzamiento.

Mantenimiento

Una vez que la fase de prueba ha concluido con éxito, se autoriza la aplicación productiva del software. La última fase del modelo en cascada incluye la entrega, el mantenimiento y la mejora del software. En esta fase se analizan los resultados del hito anterior para realizar los cambios pertinentes para dar por concluido el proyecto.

2.9.2. Ventajas y desventajas del modelo Cascada

Esta metodología permite estructurar la organización de forma clara en aquellos proyectos de desarrollo en los que las diversas fases de proyecto se diferencian claramente entre sí. Como cada una de las fases concluye con un hito, el proceso de desarrollo es muy fácil de comprender. El punto clave del modelo reside en la documentación de todos y cada uno de los pasos de proceso. Los conocimientos adquiridos se registran en pliegos de requisitos o borradores preliminares.

En teoría, el desarrollo en cascada pretende crear los requisitos previos para una ejecución rápida y rentable de los proyectos a través de una cuidada planificación previa. Sin embargo, la utilización del modelo en la práctica es controvertida. Por una parte, en el desarrollo de software las fases de proyecto no suelen estar claramente diferenciadas entre sí.

En sentido estricto, el modelo en cascada no prevé la realización de ajustes a lo largo del proyecto. Sin embargo, un proyecto de software en el que todos los detalles del desarrollo se definieran al comienzo, solo podría concluir con éxito si desde el principio se invirtiera una gran cantidad de tiempo y dinero en análisis y diseño.

En resumen se tienen las siguientes ventajas:

- Una estructura sencilla gracias a unas fases de proyecto claramente diferenciadas.
- Buena documentación del proceso de desarrollo a través de unos hitos bien definidos.
- Los costes y la carga de trabajo se pueden estimar al comenzar el proyecto.
- Aquellos proyectos que se estructuran en base al modelo en cascada se pueden representar cronológicamente de forma sencilla.

Como también las siguientes desventajas:

- Por norma general, los proyectos más complejos o de varios niveles no permiten su división en fases de proyecto claramente diferenciadas.
- Poco margen para realizar ajustes a lo largo del proyecto debido a un cambio en las exigencias.
- El usuario final no se integra en el proceso de producción hasta que no termina la programación.
- En ocasiones, los fallos solo se detectan una vez finalizado el proceso de desarrollo.

Se elige el modelo cascada dado que todas las ventajas se ajustan perfectamente al desarrollo del prototipo de sistemas de video-vigilancia. Por lo tanto las fases que tiene el modelo de desarrollo de software elegido para la realización del proyecto son:

1. **Análisis:** Esta fase se lleva a cabo a través de la definición de requisitos, la planificación y los casos de uso.
2. **Diseño:** Esta fase se lleva a cabo mediante un diagrama de secuencia y un diseño de la interfaz.

3. **Implementación:** Esta fase se lleva a cabo mediante la implementación del Sistema de video-vigilancia inteligente.
4. **Pruebas:** Esta fase se lleva a cabo mediante la realización de pruebas sobre el sistema de video-vigilancia.

Capítulo 3

Seguridad en el hogar

3.1. Introducción

La seguridad en el hogar es un tema relevante y delicado de manejar, principalmente cuando se trata del espacio vital más importante de todos, donde convive el ser humano. Sea del tipo que sea, en el hogar se establecen los vínculos más íntimos y personales; entonces la presencia de una persona en el hogar es un factor de seguridad de gran importancia. La mayoría de percances como intrusión de un extraño, allanamientos y robos se producen durante su ausencia. Los motivos para dejar un hogar vacío son varios: desde un viaje prolongado a salidas más o menos puntuales, regulares y/o diarias. Una vivienda vacía es más vulnerable que otra ocupada y este aspecto se toma en cuenta en el diseño de un sistema de seguridad para el hogar, el cual debe ofrecer características que ayuden a minimizar el impacto de las situaciones de peligro.

3.2. Ausencia en el hogar

En materia de seguridad del hogar, cualquier precaución, resulta de utilidad para prevenir, que ocurran incidentes y mantener protegidos a los seres más cercanos. De hecho el objetivo de la seguridad implica la toma de precauciones necesarias para que un lugar sea seguro para las personas. Incluso las personas más retraídas y amantes de la soledad y el aislamiento deben, en un momento u otro, salir de su residencia habitual, algo que se convierte en largas horas de ausencia en la mayoría de los casos (evidentemente por trabajo, obligaciones académicas y otros menesteres cotidianos), y ocasionalmente, con mayor o menor asiduidad, por otras razones menos frecuentes (viajes, vacaciones, etc).

Todas las posibles situaciones a presentarse se definen en función del tiempo de ausencia; de modo que se establecen distintos casos con aspectos peculiares y específicos referentes a la seguridad y los riesgos; por ejemplo exponer las medidas de protección más básicas y elementales que siempre deben tomarse en cuenta, como el contrato de un seguro para la vivienda (con elementos importantes que figuran en toda póliza para su elección y contratación).

Un seguro contra robos puede proteger de alguna manera los daños materiales que pueden ocurrir, pero incluso para probar la veracidad del suceso es necesario presentar pruebas visuales que sirvan de referencia en la denuncia. Una imagen como en la figura 3.1 puede ayudar incluso en una investigación.



Figura 3.1: Ilustración ejemplo de un intruso.

Fuente: Web

A continuación se detallan las situaciones más comunes que pueden presentarse, con sus respectivas acciones sugeridas para disminuir el peligro.

3.2.1. Ausencias cotidianas

Las ausencias diarias de horas o minutos son las más comunes y brindan oportunidades a asaltantes atentos. Para este caso se puede tener en cuenta medidas de protección sencillas y sin complicaciones que cualquiera puede llevar a cabo apenas sin inversión alguna. Asegurar los cierres de los accesos a la vivienda, disimular las ausencias o evitar proporcionar información sobre nuestros hábitos son algunas de las medidas que se exponen para evitar intrusiones no deseadas en el hogar.

3.2.2. Ausencias de término medio

Cuando uno sale de casa previamente sabe si uno va a volver al cabo de pocas horas, de unos días o de semanas; en cada caso se pueden presentar algunas peculiaridades y riesgos específicos que se deben afrontar de distintos modos. En este supuesto, tras las ausencias cotidianas, se detallan los casos de ausencias de pocos días, especialmente en fines de semana, puentes festivos y vacaciones cortas. En estas situaciones convergen la necesidad de contar con alarmas y avisadores técnicos, con la de disponer de sistemas de alarma y dispositivos antiintrusión los cuales, como veremos, pueden ser de muy diversa índole.

3.2.3. Ausencias prolongadas

Las vacaciones y las estancias de cierta duración en lugares alejados de nuestras residencias habituales ofrecen oportunidades únicas a posibles asaltantes. No ofrecer información sobre nuestro paradero, tratar de evitar el efecto de vivienda vacía, contar con la supervisión regular de alguien de confianza en nuestra ausencia y mantener a buen recaudo bienes u objetos de valor serán, en estos casos, las principales prioridades (sobre todo en el caso de las segundas residencias, una cuestión que también consideraremos detalladamente como caso diferenciado).

3.3. Situaciones de riesgo

3.3.1. Presencia de intrusos

Un intruso o persona ajena siempre representa un peligro en el interior de nuestro hogar y aún más cuando se desconoce el motivo de su presencia. La posibilidad de robos en cualquier ciudad del mundo esta presente y aun mas cuando este entra al interior de un hogar forzando cerraduras, encapuchado especialmente cuando los habitantes de la casa no estan. En la figura 3.2, se muestra una ilustracion de ejemplo de un intruso forzando la puerta de una casa.



Figura 3.2: Ilustración ejemplo de un ladrón.

Fuente: Web

3.3.2. Fuego y humo

El fuego es una reacción química, donde un conjunto de partículas o moléculas incandescentes en materia combustible es capaz de emitir calor y luz. Con el calor se pueden llegar a desintegrar muchos objetos y estos mismos servir de combustión para que el fuego se expanda. Este fenómeno es uno de los principales causantes de tragedias en la actualidad, tanto como incendios forestales y/o colectivos, incendios en interiores, como ser casas, departamentos o sitios cerrados. En la figura 3.3 se visualiza la facilidad con la que el fuego puede expandirse en interiores.



Figura 3.3: Ilustración ejemplo de fuego en interiores.

Fuente: Web

El humo acompañado del fuego son elementos muy perjudiciales tanto como a las personas como al medio ambiente en general. El humo es uno de los factores principales que afectan a la salud respiratoria de las personas y animales en general. Identificar a tiempo la presencia de humo puede incluso prevenir y/o predecir la organización de fuego evitar tragedias.

En la figura 3.4, se visualiza como la presencia de humo puede ayudar a alertar de que hay fuego en el interior de una casa o habitación.



Figura 3.4: Ilustración de la presencia de fuego y humo en una habitación cerrada.

Fuente: Web.

3.4. Sistemas de seguridad

En el mercado, existe una gran variedad de artefactos, que están al alcance de todos para proteger los hogares frente a cualquier tipo de amenaza, tanto interna como externa. Los más eficaces y eficientes, son los sistemas electrónicos de seguridad. No obstante, sea cual sea la opción elegida se debe tener en cuenta los siguientes riesgos y amenazas:

- **Allanamientos, intrusiones y vandalismo:** riesgos procedentes del exterior, que se pueden mitigar fácilmente instalando cierres de alta seguridad en los accesos a la vivienda, alarmas antiintrusión u otros mecanismos disuasorios.
- **Accidentes domésticos:** riesgos procedentes del interior de hogar que pueden poner en riesgo la integridad física y/o moral de sus habitantes, tanto personas como mascotas, así como los bienes que contienen e incluso la misma infraestructura.

Las alarmas técnicas (alertas de fugas y escapes) y de emergencia son los sistemas más adecuados para proteger una vivienda. También es preciso tomar las medidas oportunas para proteger los componentes más sensibles del hogar (instalaciones de suministros y otros elementos de riesgo) de manipulaciones indebidas, golpes y otro tipo de percances que pueden ocasionar accidentes o situaciones indeseables.

3.4.1. Alarmas

Las alarmas son artefactos sonoros que emiten un sonido que provoca la alerta en las personas. Existen de diferentes tipos, medidas y campo de uso. El volumen y el sonido es claramente diferenciado de cualquier objeto que emita un sonido cualquiera. Este objeto es utilizado generalmente para poner en alerta a todas las personas que lleguen a escucharlo y/o comunicar peligro. En la figura 3.5 se puede apreciar un modelo particular de alarmas sonoras.



Figura 3.5: Ilustración de alarmas con sonido.

Fuente: Web.

3.4.2. Sensores

Los sensores son dispositivos que captan magnitudes físicas (variaciones de luz, temperatura, sonido, etc.) u otras alteraciones de su entorno. Los detectores de humo son dispositivos desarrollados para detectar la presencia de un incendio en el interior de un edificio. En la figura 3.6 se aprecia un modelo en particular de sensor de detección de humo.



Figura 3.6: Ilustración de detector de humo.

Fuente: Web.

3.4.3. Cámaras

Las cámaras son dispositivos que permiten registrar imágenes estáticas y en movimiento. Específicamente las cámaras de vigilancia son las que se encargan de grabar todo lo que puede ocurrir en una casa o negocio. Contar con este tipo de cámara puede proporcionar sensación de seguridad y protección. Disponer de este tipo de sistemas puede resultar ser una solución para mantenerse protegido. El desarrollo de la tecnología ha logrado que el sector de la seguridad disponga de equipos eficientes y con diversas funcionalidades. En la figura 3.7 se visualizan diferentes modelos de cámaras de seguridad que se encuentran en el mercado.



Figura 3.7: Ilustración de diversas cámaras de seguridad.

Fuente: Web.

El tipo más común en el mercado son las cámaras de interiores ya que son las más sencillas y económicas del mercado ya que no necesitan mucho mecanismo ni protección. En la figura 3.8 se visualiza un ejemplo de cámara de interiores.



Figura 3.8: Cámara de vigilancia de interiores.

Fuente: Web.

Seguridad y vigilancia son aspectos que se requieren en todo el mundo; gobiernos, empresas, instituciones financieras, organizaciones de salud, necesitan cierto grado de medidas de seguridad y como resultado se generó un dramático incremento en la demanda de aplicaciones de seguridad como por ejemplo video vigilancia, monitoreo y grabación de: fronteras, puertos, transporte, hogares, corporaciones, instituciones educativas, lugares públicos, edificios, etc.

Capítulo 4

Análisis y Diseño

En las etapas iniciales de la metodología “Cascada”, se realiza el análisis del problema y diseño previo del sistema, todo esto al inicio de la implementación del producto final. Es importante que estas fases sean completadas en su totalidad para que el desarrollo sea exitoso. Este proyecto plantea el diseñado y desarrollo para 70 días hábiles según calendario, con la distribución del tiempo definido según la complejidad de las tareas que implican cada una de las fases.

4.1. Análisis

Para el diseño del Sistema de Video-Vigilancia Inteligente, es necesario tener en cuenta los elementos principales que lo componen. Un sistema de video-vigilancia esta compuesto de cámaras individuales y un puesto central o servidor donde todas las conexiones convergen y se centralizan para su control. En el dispositivo central (servidor) se procesan las imágenes que las cámaras capturan y se convierten en video para ser visualizado en un monitor.

El sistema propuesto permite visualizar video en vivo desde cualquier dispositivo con acceso a la red de internet además de que envía notificaciones automáticas en el instante en que se detecta: movimiento, fuego, o silueta de un intruso. El usuario recibe la notificación por correo electrónico el cual adjunta capturas y un enlace web para visualizar en vivo lo que esta captando la cámara.

En la figura 4.1 se visualiza el esquema general del sistema propuesto. Las cámaras de video-vigilancia se encargan de capturar los fotogramas de video y se enlazan por medio de un socket o conector (uno por cada cámara) al servidor central. Cuando es registrada una nueva conexión, el sistema notifica al usuario por medio de un correo electrónico, compartiendo información relevante sobre la conexión de una nueva cámara. El servidor central, maneja todas las conexiones, además realiza el análisis de los fotogramas de manera individual por cada cámara conectada por medio de una librería de visión por computadora. De forma paralela se construye el video que se va a transmitir a partir de los fotogramas, siendo decodificado por medio de un conjunto de paquetes de software libre, los cuales arman las partes del archivo para el streaming de video por medio de la red. Cuando un evento (fuego, movimiento, silueta humana) es identificado, automáticamente el sistema envia la notificación correspondiente por medio de correo electrónico al usuario avisando una posible situación identificada.

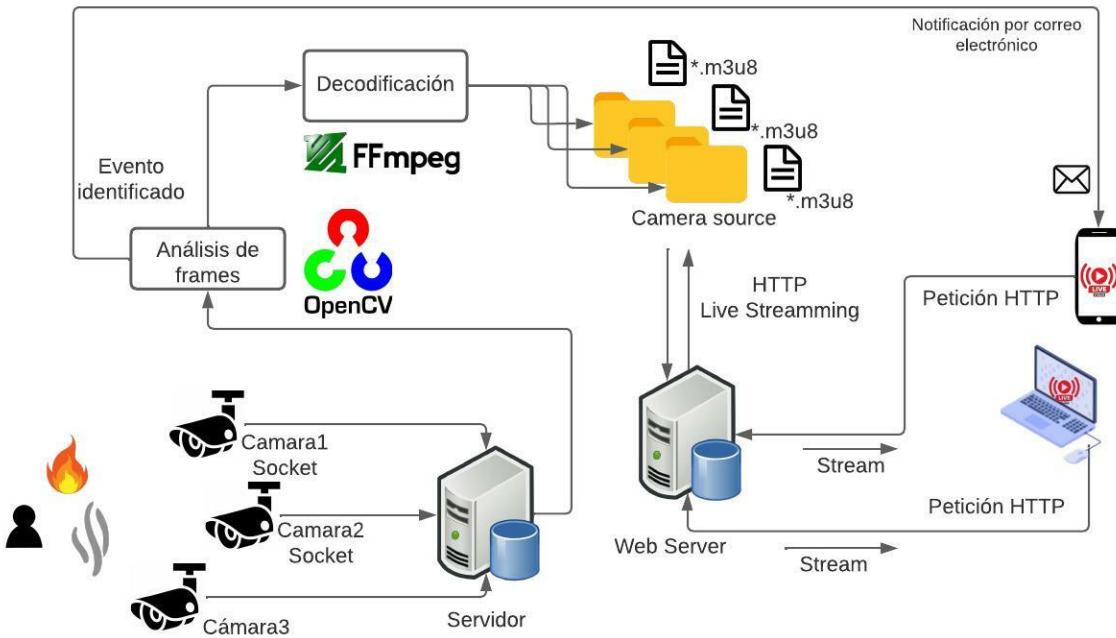


Figura 4.1: Diseño de interacción de los módulos del sistema de video-vigilancia.

Fuente : Elaboración propia

4.1.1. Definición de Requerimientos

La definición de requerimientos es una de las actividades más importantes del desarrollo de software; de ello depende el resto de actividades del proyecto. Anteriormente se describe completamente el comportamiento del sistema que se desarrolla para la identificación de los requerimientos. Inicialmente se plantean los criterios de partida a tomar en cuenta en el diseño del sistema de video-vigilancia inteligente:

- Costos altos en la infraestructura de transmisión de video en vivo.
- Las características de identificación automática, estan disponibles para sistemas de video-vigilancia de alto nivel.
- Una alerta inmediata puede minimizar el impacto de alguna situación que ponga en peligro la integridad de bienes materiales y humanos.
- Los usuarios finales son personas que a menudo dejan su hogar para salir a trabajar, y usan constantemente su correo electrónico.

Requerimientos Funcionales

En esta fase es necesario delimitar el alcance y las capacidades del sistema planteado para la realización de la planificación inicial, estimación de tiempos, diseño y desarrollo. Para ello se define la lista de requerimientos funcionales del sistema de video-vigilancia inteligente.

Tabla 4.1: Lista de requerimientos funcionales

| | |
|-----|--|
| 1. | Capturar fotogramas por medio de una o varias cámaras portátiles. |
| 2. | Visualizar fotogramas capturados en tiempo real (captura de video). |
| 3. | Notificar al usuario cuando una nueva cámara se conecta. |
| 4. | Notificar al usuario cuando una cámara se desconecta. |
| 5. | Enviar fotogramas capturados por medio de la red al servicio encargado de su análisis. |
| 6. | Analizar y procesar fotogramas capturados individualmente por cada cámara. |
| 7. | Permitir la recepción de fotogramas de varias fuentes hacia el servicio. |
| 8. | Transmitir los fotogramas convertidos en video en vivo desde fuentes diferentes. |
| 9. | Detectar movimiento a partir de fotogramas recibidos desde distintas fuentes. |
| 10. | Detectar silueta humana a partir de fotogramas recibidos desde distintas fuentes. |
| 11. | Detectar fuego a partir de fotogramas recibidos desde distintas fuentes. |
| 12. | Notificar al usuario por medio de un correo electrónico cuando se de una detección. |

Fuente: Elaboración propia.

Requerimientos No Funcionales

Los requerimientos no funcionales son aquellos relacionados con la calidad y el proceso de desarrollo del sistema. Los tipos de requisitos no funcionales aplican: rendimiento, disponibilidad, accesibilidad, usabilidad, estabilidad, portabilidad, costo, operatividad, interoperabilidad, escalabilidad, concurrencia, mantenibilidad, interfaz, plazo de entrega y herramientas. Los requisitos no funcionales surgen de la necesidad del usuario, debido a las restricciones en el presupuesto y las herramientas utilizadas.

1. Requisitos de interfaz

- Las cámaras tendrán una interfaz que permita visualizar la tarea que ejecutan.
- El servidor mostrará información necesaria sobre las tareas que realiza.
- El sistema deberá ser de fácil configuración.
- Las notificaciones por correo electrónico deberán ser visualmente estilizadas.

2. Requisitos de portabilidad

- Los módulos de cámara podrán conectarse de forma cableada o inalámbrica.

3. Requisitos de disponibilidad

- La transmisión en vivo estará disponible cuando el usuario quiera visualizar lo que captan las cámaras.

4.2. Planificación

Todas las fases del modelo cascada son planificadas según la complejidad y tareas que presenta cada fase. Dado que cada fase debe culminarse por completo para pasar a la siguiente o en su

defecto requerir mínimas modificaciones para regresar a la fase anterior, se plantea una planificación de 70 días hábiles que se visualiza en la tabla 4.2.

Tabla 4.2: Tabla de planificación según fases del modelo Cascada.

| Num. | Fase del modelo Cascada | Fecha inicial | Fecha final | Duración (días) |
|------|----------------------------|---------------|--------------|-----------------|
| 1. | Fase de análisis | 6-jun | 17-jun | 10 |
| 2. | Fase de diseño del sistema | 20-jun | 8-jul | 15 |
| 3. | Fase de implementación | 11-jul | 19-agosto | 30 |
| 4. | Fase de pruebas | 22-agosto | 2-septiembre | 10 |
| 5. | Fase de mantenimiento | 5-septiembre | 9-septiembre | 5 |

Fuente: Elaboración propia.

En la figura 4.2 se visualiza e diagrama de Gannt en base a la planificación expresada en la anterior tabla.

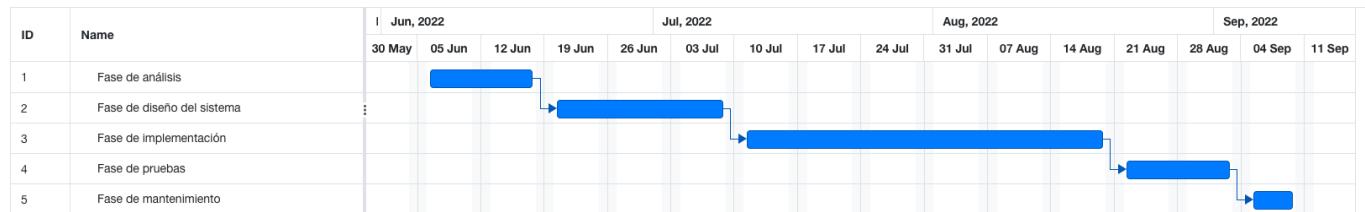


Figura 4.2: Diagrama de Gannt.

Fuente : Elaboración propia

4.3. Diseño

El diseño del sistema de video-vigilancia se divide y se desarrolla según los módulos que componen el sistema completo:

- Módulo de cámaras
- Módulo de servidor

4.3.1. Módulo de cámaras

El módulo de cámaras permite el control de conexión de una cámara al servidor central. Permite conectar una cámara web o una cámara de una Raspberry Pi. Presenta campos para ingresar datos de configuración que permiten la conexión al servidor central.

Diseño de interfaz

En la figura 4.3 se presenta el esbozo de interfaz de usuario que servirá de diseño final para el desarrollo del presente módulo. El módulo de cámaras permite la visualización de fotogramas

capturados por medio de la cámara conectada, configuración de la conexión y verificación del estado de envío de los fotogramas al servidor.

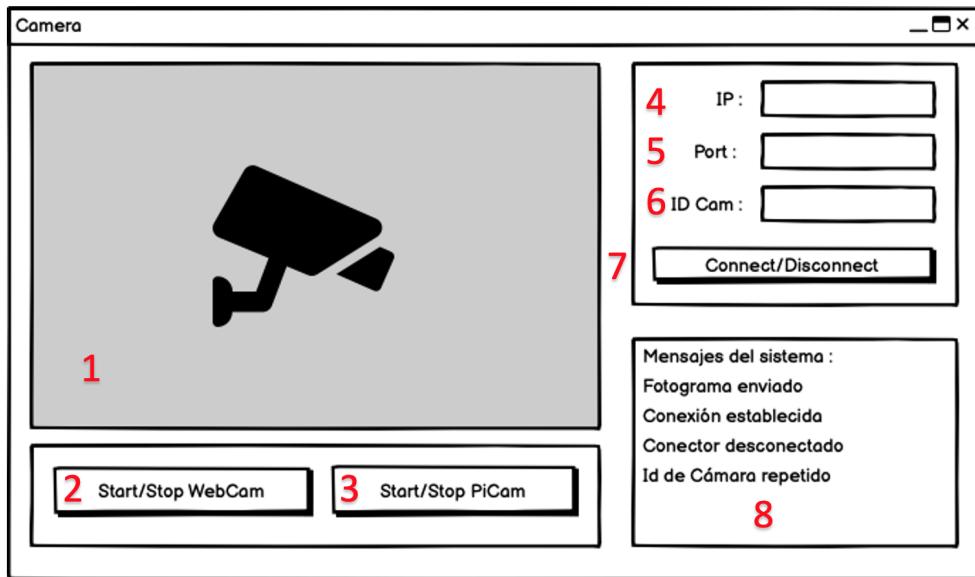


Figura 4.3: Diseño de interfaz gráfica de usuario (Módulo de cámaras).

Fuente: Elaboración propia.

A continuación se describe a detalle cada uno de los elementos que conforman la interfaz de usuario del módulo de cámaras.

1. **Pantalla:** Espacio de visualización de los fotogramas que capta la cámara conectada.
2. **Botón WebCam:** Botón de conexión/desconexión para la cámara web
3. **Botón PiCam:** Botón de conexión/desconexión para la cámara de Raspberry Pi.
4. **Campo de dirección IP:** Campo de texto para que el usuario ingrese la dirección IP del servidor.
5. **Campo de número de puerto:** Campo de texto para que el usuario ingrese el puerto de escucha del servidor.
6. **Campo de identificador de la cámara:** Campo de texto para que el usuario ingrese el número identificador de la cámara conectada.
7. **Botón de conexión:** Botón de conexión/desconexión al servidor para el envío de fotogramas al servidor.
8. **Cuadro de mensajes:** Cuadro de lista para mostrar diferentes mensajes de módulo: envío de fotogramas, conexión/desconexión al servidor, etc.

Diagrama de secuencia

En la figura 4.4 se ilustra el diagrama de secuencia del módulo de cámaras. El diagrama describe de forma gráfica el comportamiento de la interacción entre el módulo de cámaras con el servidor.

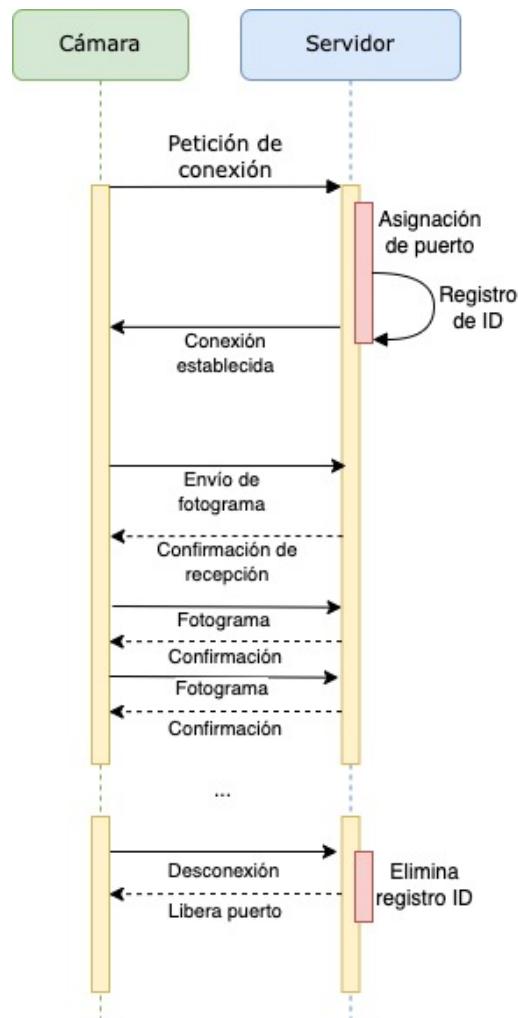


Figura 4.4: Diagrama de interacción entre el módulo de cámaras con el servidor.

Fuente: Elaboración propia.

La interacción entre el módulo cámaras y el servidor es como sigue:

1. El módulo de cámaras realiza una petición de conexión al servidor.
2. El servidor acepta la petición y le asigna un puerto de escucha a esa nueva conexión.
3. Se registra el identificador de la cámara para evitar duplicidad en los identificadores.
4. El servidor envía un mensaje de confirmación de conexión.
5. El módulo de cámaras envía un fotograma.
6. El servidor confirma la recepción del fotograma.

7. Se repiten los anteriores pasos mientras dura la conexión.
8. El módulo de cámaras se desconecta del conector asignado.
9. El servidor elimina el registro del identificador de la cámara y libera la conexión.

Diagrama de clases

De acuerdo al planteamiento de comportamiento del módulo de cámaras, para el desarrollo de este módulo se plantea el diagrama de clases que se visualiza en la figura 4.5.

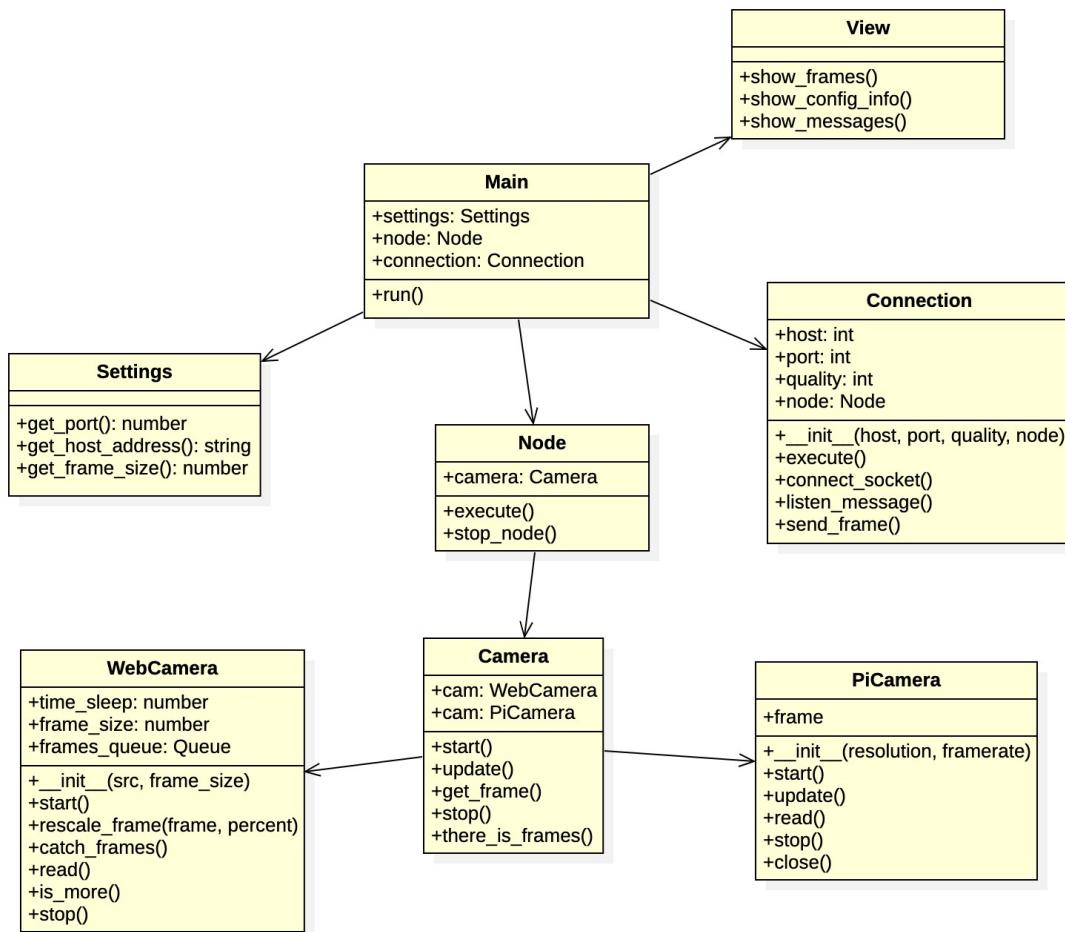


Figura 4.5: Diagrama de clases del módulo de cámaras.

Fuente : Elaboración propia

Para la comprensión de este diagrama de clases se procede a describir cada clase con sus respectivos atributos y métodos.

- **Main:** Clase principal encargada de inicializar las instancias necesarias para el módulo de cámaras. Sus atributos son:

- **settings:** Instancia de la clase Settings, que permite cargar las configuraciones iniciales para iniciar el módulo.
- **node:** Instancia de la clase Node, que representa el ciclo de petición de los fotogramas de la cámara.
- **connection:** Instancia de la clase Connection, que permite la conexión/desconexión con el servidor a partir de la configuración por defecto establecida en el archivo de configuración inicial.

Los métodos de clase son:

- **run:** Método encargado de instanciar la clase View que representa la interfaz de usuario.
- **Node:** Clase encargada de implementar los métodos que permiten obtener los fotogramas directamente cual se la cámara conectada (WebCam o PiCam), tambien envia los fotogramas tanto al servidor y a la interfaz de usuario para su visualización. Este es su único atributo:
 - **camera:** Es la instancia de la cámara que permite la captura de fotogramas.

Los métodos de clase son:

- **execute:** Método encargado de ejecutar la gestión de fotogramas capturados por medio de la cámara física.
- **stop_node:** Se encarga cerrar el proceso de gestión de fotogramas al momento de desconectarse del servidor.
- **Settings:** Clase encargada de cargar las configuraciones des el archivo inicial de configuración para mostrarlo en pantalla. Los métodos de clase son:
 - **get_port:** Obtiene el valor del puerto del servidor desde el archivo de configuración.
 - **get_host_address:** Obtiene la dirección del servidor desde el archivo de configuración.
 - **get_frame_size:** Obtiene el valor del tamaño del fotograma según la configuración.
- **Connection:** Clase encargada de gestionar los fotogramas obtenidos por medio de la cámara. Los atributos de la clase son:
 - **host:** Dirección del servidor.
 - **port:** Puerto por defecto para conectarse al servidor.
 - **quality:** Valor de referencia para la transformación del fotograma a ser enviado.
 - **node:** Instancia de la clase Node que permite gestionar los fotogramas para ser mostrados y enviados.

Los métodos de la clase son:

- **execute:** Permite la ejecución en paralelo de la conexión.
- **connect_socket:** Permite conectarse al servidor por medio de un conector.
- **listen_message:** Método encargado de recibir mensajes por parte del servidor en el momento de la conexión y desconexión del módulo de cámaras.

- send_frame: Método encargado de enviar los fotogramas por medio de un conector hacia el servidor.
- **Camera:** Clase encargada de representar la cámara genérica que es conectada físicamente al módulo. Estos son sus atributos:
- web_camera: Instancia de la clase Webcamera, que representa una cámara web.
 - pi_camera: Instancia de la clase PiCamera, que representa una cámara de Raspberry Pi.
- Estos son los métodos de clase:
- start: Método de interfaz que incializa la cámara.
 - get_frame: Método que devuelve el frame actual que recibe de la cámara física.
 - stop: Detiene el proces de captura de fotogramas.
 - there_is_frames: Devuelve un valor verdadero y falso de acuerdo a si hay fotogramas almacenados en memoria o no.
- **WebCamera** Clase que representa una cámara web. Estos son sus métodos:
- start: Inicializa la cámara web despues de ser conectada.
 - rescale_frame: Redimensiona la imagen para poder ser visualizada en la interfaz de usuario.
 - catch_frames: Captura la imagen por medio del lente de la cámara.
 - stop: Detiene el proceso de captura de fotogramas por medio de la cámara web.
- **PiCamera** Clase que representa un cámara de Raspberry Pi. Estos son sus métodos:
- start: Inicializa la cámara web despues de ser conectada.
 - read: Método encargado de capturar un fotograma a partir del lente de la cámara.
 - stop: Método que detiene la captura de fotogramas.
 - close: Cierra el proceso de captura de fotogramas por medio de la cámara de Raspberry pi.

4.3.2. Módulo de servidor

El servidor que se encarga de la transmision

Diagrama de secuencia

Se describe el proceso entre el servidor y la cámara para el respectivo análisis de los frames

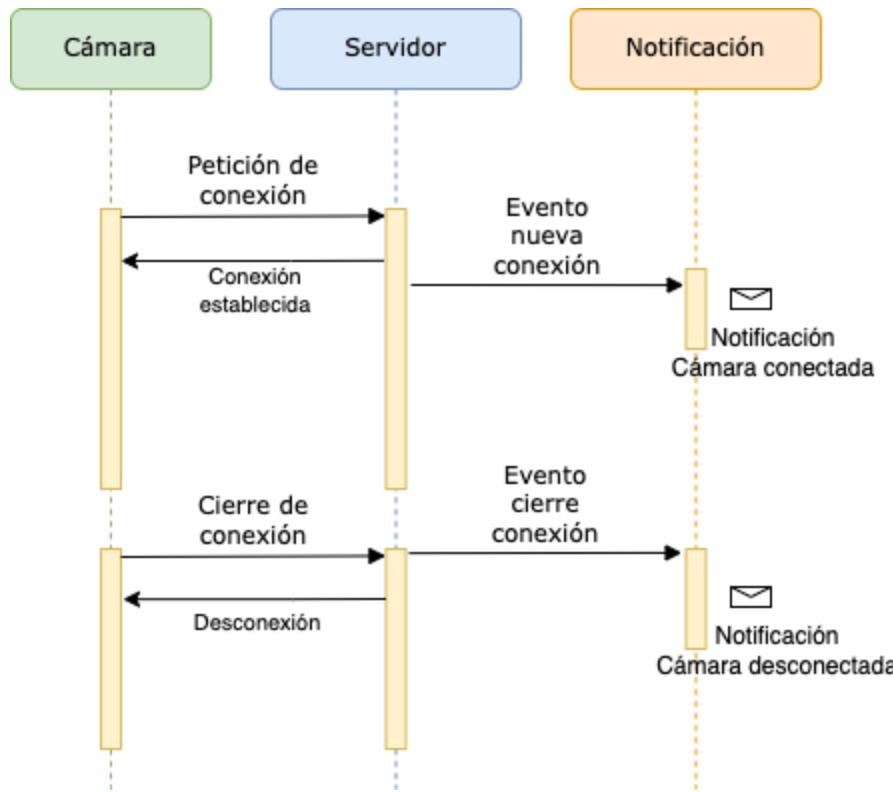


Figura 4.6: Diagrama de secuencia: Notificación de conexión/desconexión.

Fuente: Elaboración propia.

A continuación se describe la interacción entre la cámara, el servidor y el proceso de notificación:

1. El módulo de cámaras realiza una petición de conexión al servidor.
2. El servidor responde la petición aceptando la conexión con la designación un puerto.
3. El servidor registra el identificador, hora y fecha de conexión de la cámara.
4. El servidor prepara la notificación con los datos registrados e información sobre otras cámaras disponibles.
5. El servidor envia notificación por correo electrónico a la dirección configurada.
6. El servidor prepara el correo electrónico con los datos registrados, información sobre otras cámaras disponibles anteriormente y se envia a la dirección configurada.
7. El módulo de cámaras se desconecta y se registra fecha y hora de la desconexión.
8. El servidor prepara la notificación con la fecha y hora de desconexión con información adicional de cámaras disponibles.
9. El servidor envia la notificación por correo electrónico.

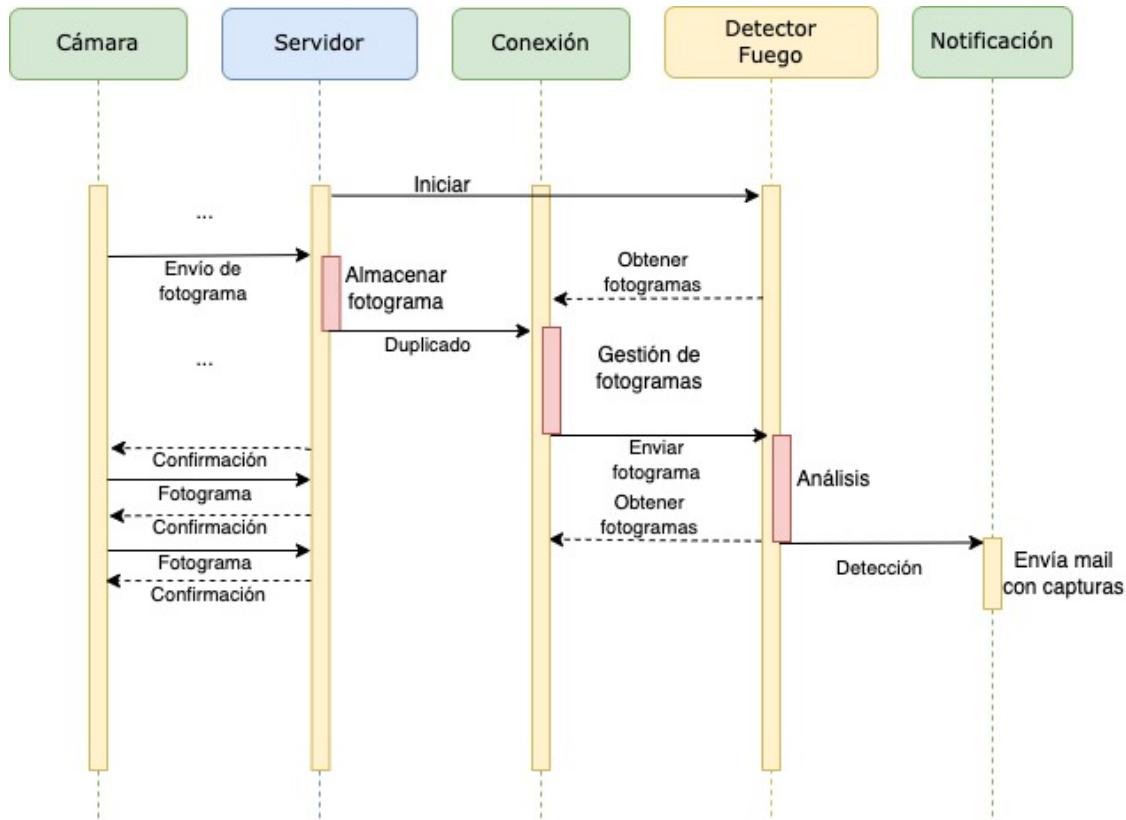


Figura 4.7: Diagrama de secuencia: Detector de fuego.

Fuente: Elaboración propia.

A continuación se describe el diagrama de secuencia de la notificación de identificación de fuego.

1. El servidor inicializa el detector de fuego.
2. El módulo de cámaras envia un fotograma.
3. El servidor recibe el fotograma, almacena en memoria y duplica para el detector de fuego.
4. El detector requiere un fotograma para su análisis.
5. El detector de fuego obtiene el fotograma y aplica el análisis.
6. Si existe una incidencia en el fotograma se almacena como posible evento.
7. El detector continua obteniendo fotogramas, analizando cada uno.
8. Cuando hay un numero determinado de incidencias en los fotogramas se prepara la notificación.
9. Se envia la notificación por medio de correo electrónico con capturas adjuntas de los fotogramas con incidencias.

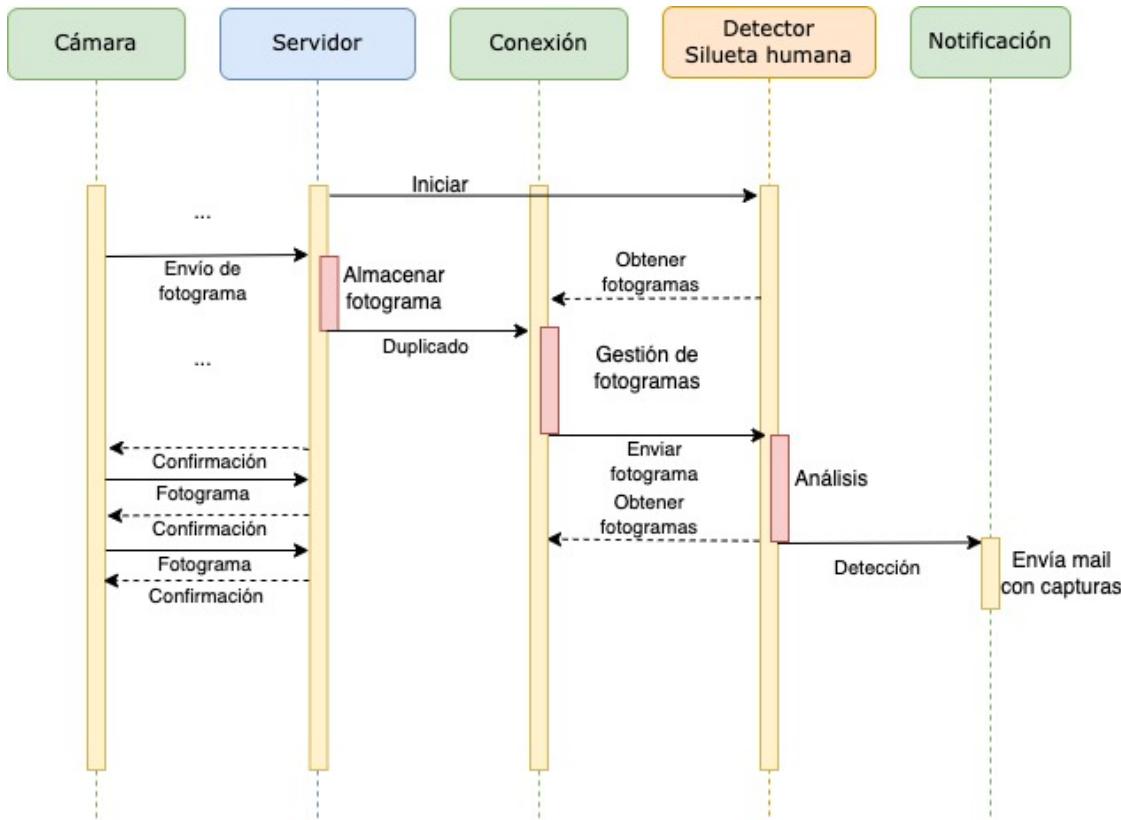


Figura 4.8: Diagrama de secuencia: Detector de silueta humana.

Fuente: Elaboración propia.

A continuación se describe el diagrama de secuencia de la notificación de identificación de silueta humana.

1. El servidor inicializa el detector de silueta humana.
2. El módulo de cámaras envia un fotograma.
3. El servidor recibe el fotograma, almacena en memoria y duplica para el detector.
4. El detector requiere un fotograma para su análisis.
5. El detector de silueta humana obtiene el fotograma y aplica el análisis.
6. Si existe una incidencia en el fotograma se almacena como posible evento.
7. El detector continua obteniendo fotogramas, analizando cada uno.
8. Cuando hay un número determinado de incidencias en los fotogramas se prepara la notificación.
9. Se envia la notificación por medio de correo electrónico con capturas adjuntas de los fotogramas con incidencias.

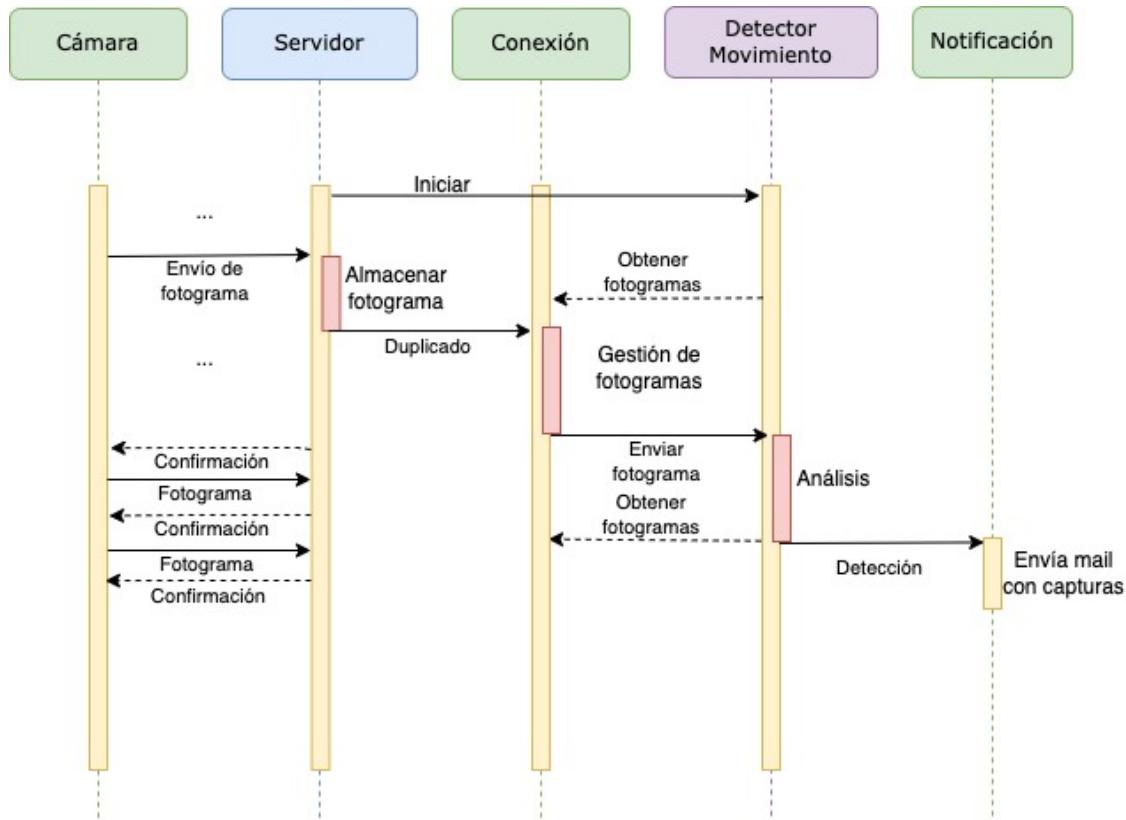


Figura 4.9: Diagrama de secuencia: Detector de movimiento.

Fuente: Elaboración propia.

A continuación se describe el diagrama de secuencia de la notificación de identificación de movimiento.

1. El servidor inicializa el detector de movimiento.
2. El módulo de cámaras envia un fotograma.
3. El servidor recibe el fotograma, almacena en memoria y duplica para el detector de movimiento.
4. El detector requiere un fotograma para su análisis.
5. El detector de fuego obtiene el fotograma y aplica el análisis.
6. Si existe una incidencia en el fotograma se almacena como posible evento.
7. El detector continua obteniendo fotogramas, analizando cada uno.
8. Cuando hay un numero determinado de incidencias en los fotogramas se prepara la notificación.
9. Se envia la notificación por medio de correo electrónico con capturas adjuntas de los fotogramas con incidencias.

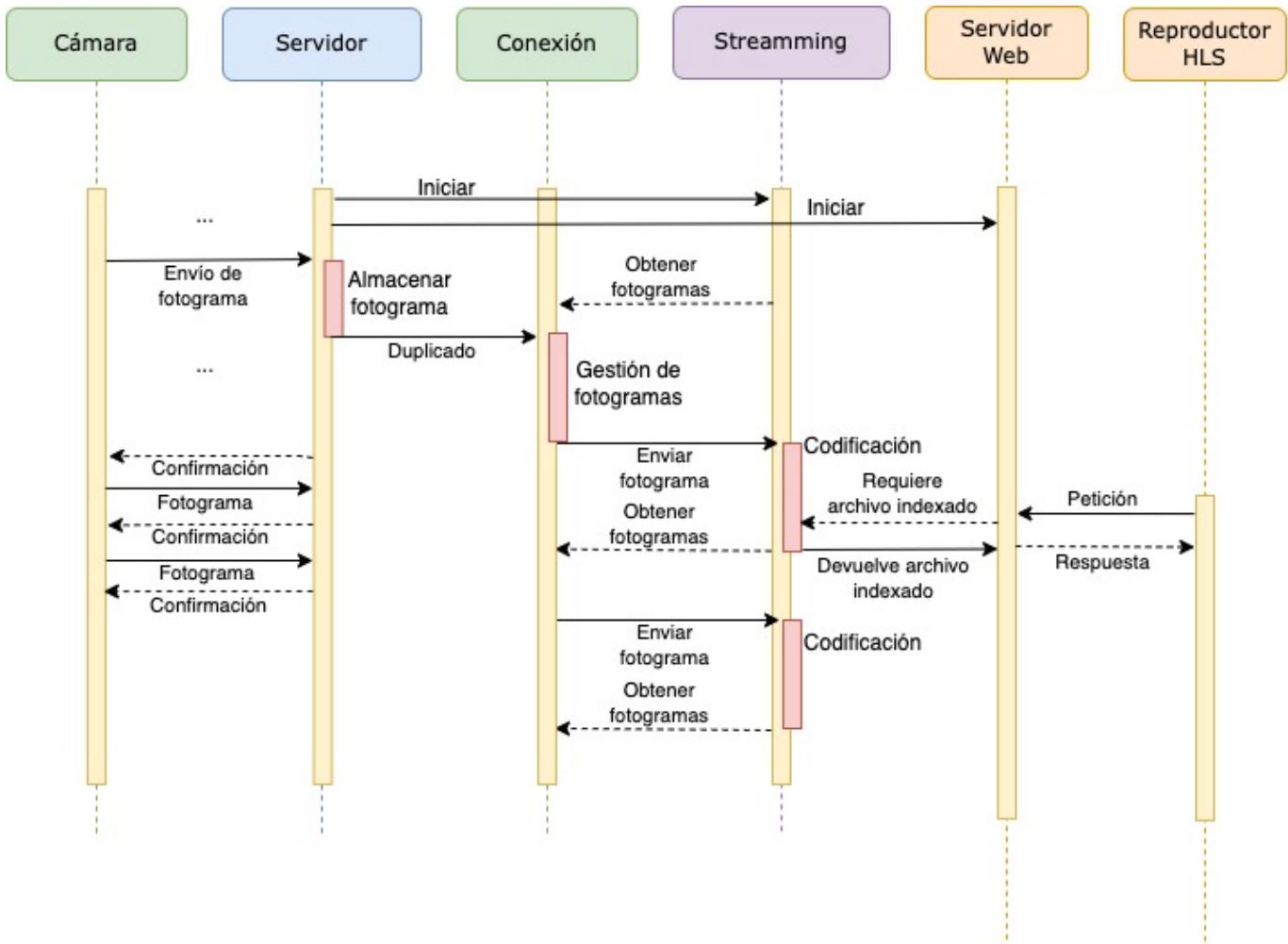


Figura 4.10: Diagrama de secuencia: Streaming de video en vivo.

Fuente: Elaboración propia.

A continuación se describe el diagrama de secuencia de la transmisión en vivo del video generado por el módulo de cámaras.

1. El servidor inicia el servicio de Streaming.
2. El servidor inicia el servidor web para la reproducción desde el navegador.
3. El módulo de cámaras envía los fotogramas hacia el servidor.
4. El servidor almacena los fotogramas y gestiona los duplicados.
5. El servicio de streaming, obtiene los fotogramas para codificar el video y crear la lista de reproducción de los fragmentos de video que son creados.
6. El servidor web está a la espera de peticiones cuando se acceda a la dirección específica de la transmisión.
7. La reproducción en vivo comienza cuando se accede a la dirección y el reproductor realiza las peticiones al servidor web.

Diagrama de clases

En la figura 4.11 se muestra el diagrama de clases del servidor.

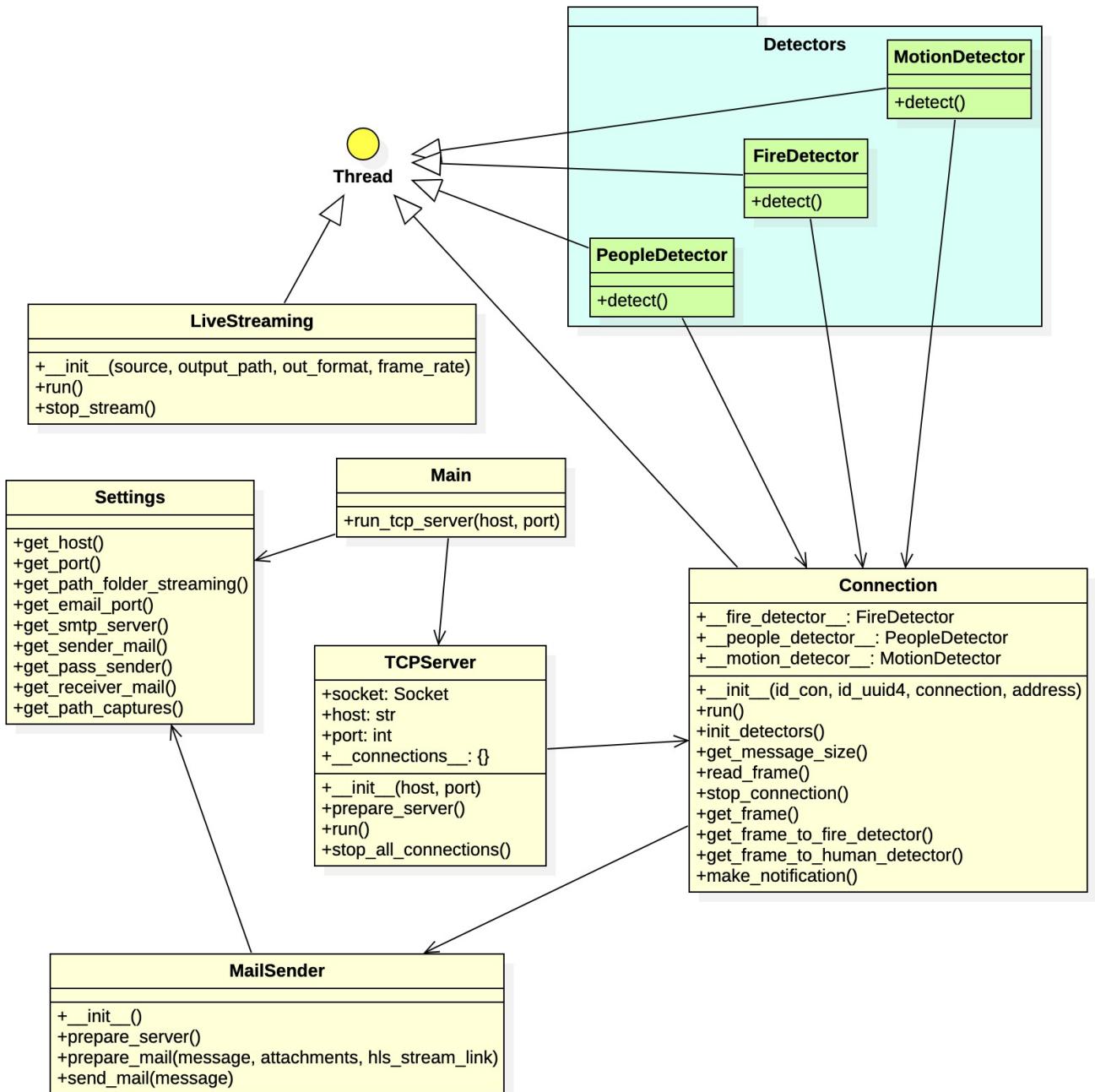


Figura 4.11: Diagrama de clases: Servidor
Fuente: Elaboración propia.

- **Main**: Es la clase principal que permite la ejecución constante del servidor. El único método de clase es:

- `run_tcp_server`: Inicializa el servidor en la dirección y puerto provistas desde la configuración.
- **TCPServer**: Clase que maneja las peticiones de nuevas conexiones del módulos de cámaras, prepara el servidor y registra los nuevos procesos que se crean a partir de una nueva conexión. Los atributos de clase son:
 - `socket`: Es la librería que permite las conexiones a sistemas externos.
 - `host`: Representa la dirección de escucha del servidor.
 - `port`: Representa el puerto general de escucha del servidor.
 - `connections`: Almacena los nuevos hilos de conexión cuando una nueva cámara se conecta.
- Los métodos de clase son:
 - `prepare_server`: Método encargado de preparar las configuraciones de l servidor antes de empezar su ejecución.
 - `run`: Lanza la ejecución del servidor.
 - `stop_all_connections`: Cierra todas las conexiones existentes en el momento de cerrar el servidor.
- **Settings**: Clase que se encarga de gestionar todos los valores de configuración en el servidor. Estos son sus métodos de clase:
 - `get_host`: Obtiene la dirección por defecto desde el archivo de configuración.
 - `get_port`: Obtiene el puerto por defecto desde el archivo de configuración.
 - `get_path_folder_streaming`: Obtiene el directorio por defecto para el streamming de video de las cámaras desde el archivo de configuración.
 - `get_email_port`: Obtiene el puerto del servicio de correo electrónico desde el archivo de configuración.
 - `get_smtp_server`: Obtiene la dirección por defecto del servicio de correo electrónico desde el archivo de configuración.
 - `get_sender_mail`: Obtiene a dirección de correo electrónico para el envío de correos de notificación.
 - `get_pass_sender`: Obtiene la contraseña del correo electrónico para el envío de correos de notificación.
 - `get_receiver_mail`: Obtiene la dirección de correo electrónico destino.
 - `get_path_captures`: Obtiene el directorio por defecto de las capturas de los detectores desde el archivo de configuración.
- **Connection**: Clase encargada de gestionar, duplicar y almacenar los fotogramas que son enviados desde el módulo de cámaras. Sus propiedades son:
 - `fire_detector`: Instancia de la clase del detector de fuego.
 - `people_detector`: Instancia de la clase del detector de silueta humana.

- motion_detector: Instancia de la clase del detector de movimiento.

Sus métodos de clase son:

- run: Ejecuta el proceso de escucha y gestión de nuevas conexiones por parte del módulo de cámaras.
 - init_detectors: Lanza los procesos de detección.
 - get_message_size: Obtiene el valor del tamaño de los mensajes para procesar las capturas recibidas por el módulo de cámaras.
 - read_frame: Recibe el fotograma y convierte la información de un valor binario a imagen
 - stop_connection: Cierre el proceso de conexión del módulo de cámaras y procesos de detección.
 - get_frame: Devuelve un fotograma hacia los detectores o al módulo de transmisión de video en vivo.
 - store_frame: Almacena en memoria un fotograma.
 - make_notification: Prepara y envía la notificación a partir de la señal de un detector.
- **LiveStreamming**: Clase que se encarga de la codificación y decodificación de video, construyendo el archivo que registra los fragmentos de video creados para su transmisión:
- init: Inicializa el módulo de transmisión de video en vivo.
 - run: Ejecuta de manera constante el decodificado del video.
 - stop_stream: Termina la transmisión y cierra el proceso de codificación.
- **MailSender**: Clase encargada de ofrecer el servicio de envío de mensajes por correo electrónico:
- init: Inicia el módulo de envío de correo electrónico.
 - prepare_server: Prepara el servicio de correo electrónico a partir de los valores de configuración.
 - prepare_mail: Prepara el mensaje de correo a partir del detector que genera la alarma.
 - send_mail: Envía el correo electrónico a la dirección registrada en el archivo de configuraciones.
- **PeopleDetector** Clase encargada de la detección de siluetas humanas en los fotogramas. Este es el único método de clase:
- detect: Método que se encarga de aplicar el algoritmo de detección de siluetas humanas sobre los fotogramas.
- **FireDetector** Clase encargada de la detección de fuego en los fotogramas. Este es el único método de clase:
- detect: Método que se encarga de aplicar el algoritmo de detección de fuego sobre los fotogramas.

- **MotionDetector** Clase encargada de la detección de movimiento en los fotogramas. Este es el único método de clase:

- detect : Método que se encarga de aplicar el algoritmo de detección de movimiento sobre los fotogramas.

Diseño de notificación por correo electrónico

Cuando se identifica un evento por parte de alguno de los tres detectores implementados, es necesario notificar al usuario; el medio elegido es un correo electrónico que debe ser más que solo un texto referente a la identificación de un suceso. Se diseña un mensaje con texto enrriquecido adjuntando mayor información en el momento de:

- Conectar una cámara del módulo de cámaras al servidor.
- Desconectar una cámara del módulo de cámaras al servidor.
- Identificación de fuego en fotogramas.
- Identificación de silueta humana en los fotogramas.
- Identificación de movimiento en los fotogramas.

Además de agregar información sobre la fecha y hora de conexión/desconexión o identificación de algún suceso; se adjunta capturas acorde a la identificación. A continuación se detalla cada uno de los ejemplos de notificación a desarrollar.

- En la figura 4.12, se detalla la conexión de una cámara y se muestra información de otras cámaras disponibles.
- En la figura 4.13, se detalla la desconexión de una cámara y no existen cámaras disponibles.
- En la figura 4.14, se detalla la desconexión de una cámara y existen otras cámaras disponibles.
- En la figura 4.15, se detalla la conexión de una cámara y no existen otras cámaras disponibles.
- En la figura 4.16, se muestra el ejemplo de notificación de identificación de fuego.
- En la figura 4.17, se muestra el ejemplo de notificación de identificación de silueta humana.
- En la figura 4.18, se muestra el ejemplo de notificación de identificación de movimiento.

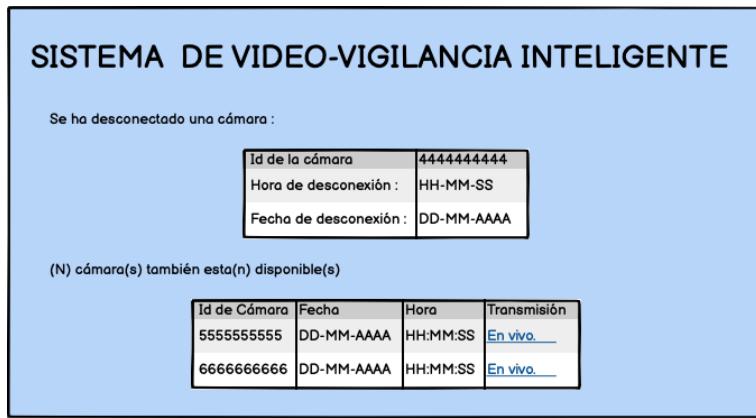


Figura 4.12: Diseño de notificación: Conecta una cámara más al sistema.
Fuente: Elaboración propia.

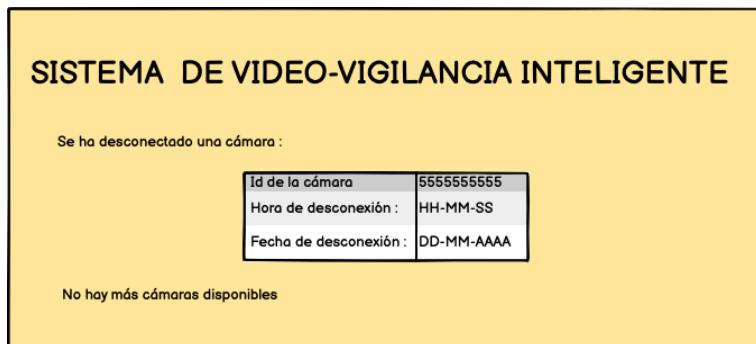


Figura 4.13: Diseño de notificación: Desconexión de la única cámara.
Fuente: Elaboración propia.

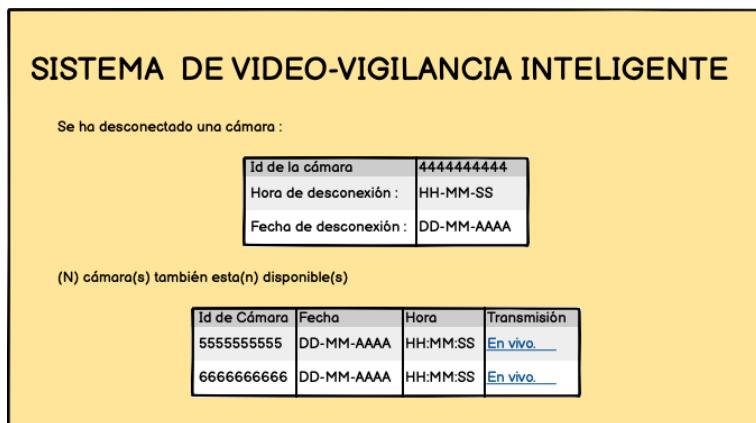


Figura 4.14: Diseño de notificación: Desconexión de una de las cámaras disponibles.
Fuente: Elaboración propia.

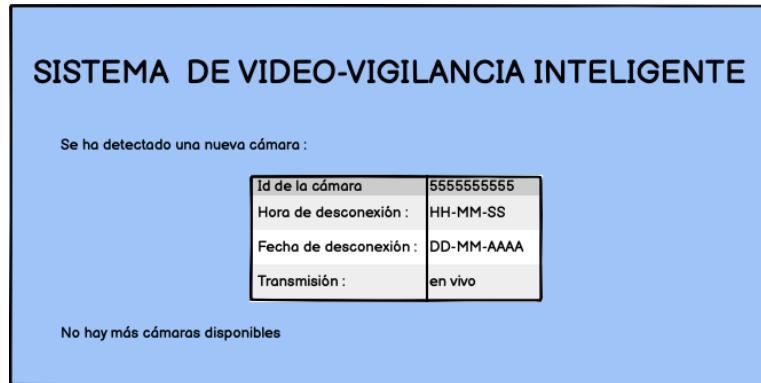


Figura 4.15: Diseño de notificación: Nueva y única conexión.
Fuente: Elaboración propia.

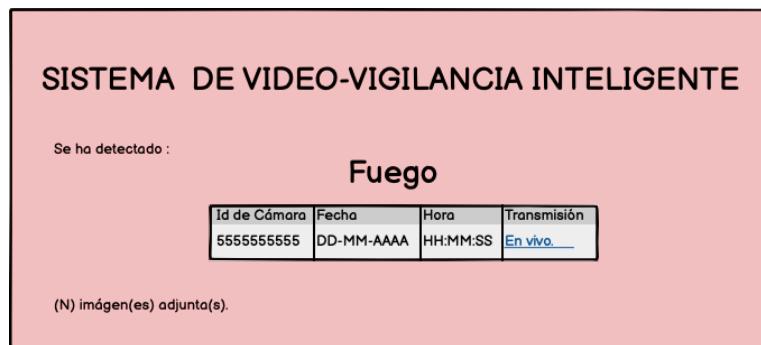


Figura 4.16: Diseño de notificación: Fuego detectado.
Fuente: Elaboración propia.



Figura 4.17: Diseño de notificación: Silueta humana detectada.
Fuente: Elaboración propia.



Figura 4.18: Diseño de notificación: Movimiento detectado.

Fuente: Elaboración propia.

Capítulo 5

Implementación

En el modelo “cascada” la fase de implementación es la más importante, debido a que en esta se desarrolla el producto final a partir del diseño de la fase anterior. El diseño previo permite al programador implementar con éxito los diferentes módulos del sistema de video-vigilancia planteado. El desarrollo de todo el sistema esta programado con el lenguaje de programación Python en su totalidad.

5.1. Módulo de Cámaras

A partir del diseño inicial se implementa la interfaz de usuario por medio de la librería “tkinter” propia del lenguaje. En la figura 5.1, se visualiza la implementación final en base al diseño de la fase anterior de la metodología.

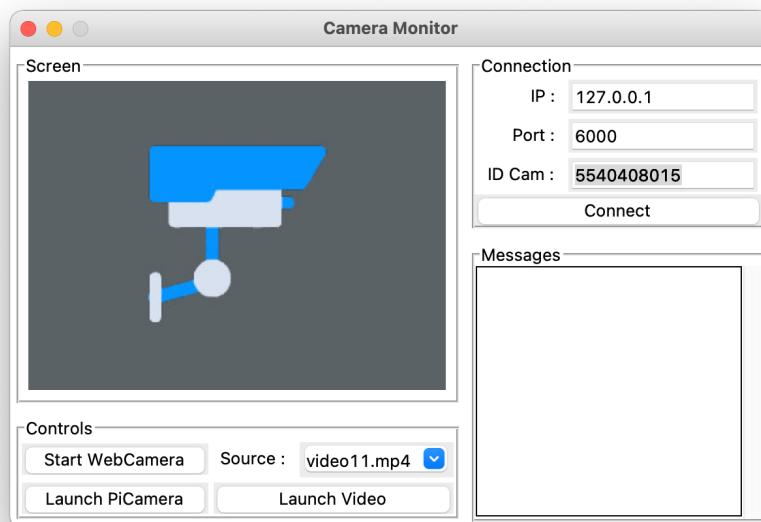


Figura 5.1: Interfaz gráfica del módulo de cámaras.
Fuente : Elaboración Propia.

La conexión entre el módulo de cámaras y el servidor se realiza por medio de otro paquete propio

del lenguaje denominado “sockets”, que es utilizado para el envío de mensajes por medio de la red. En la figura 5.2 se visualiza el código necesario para el envío de un fotograma al servidor.

```
1 def send_frame(self, frame, quality):
2     encode_param = [int(cv2.IMWRITE_JPEG_QUALITY), quality]
3     result, image = cv2.imencode('.jpg', frame, encode_param)
4     data = pickle.dumps(image, 0)
5     if self.is_connected():
6         try:
7             self._socket_connected_.\
8                 sendall(struct.pack(">L", len(data)) + data)
9         except socket.error:
10             message = self.receive_message_connection_rejected()
11             self._controller_.disconnect_to_server()
```

Figura 5.2: Código de envío de un fotograma por medio de sockets.

Fuente : Elaboración Propia.

Para que el fotograma sea enviado por la red este debe ser serializado y convertido en un paquete de datos binario y esto es logrado con el uso de los paquetes pickle y struct en las líneas 5 y 9 respectivamente.

Cuando el módulo de cámaras y el servidor se conectan, si inicia el proceso de envío de fotogramas hacia el servidor, y este es visualizado y numerado por cada fotograma enviado, en el campo de mensajes del sistema propio de la interfaz gráfica del módulo de cámaras. Este comportamiento se visualiza en la figura 5.3.

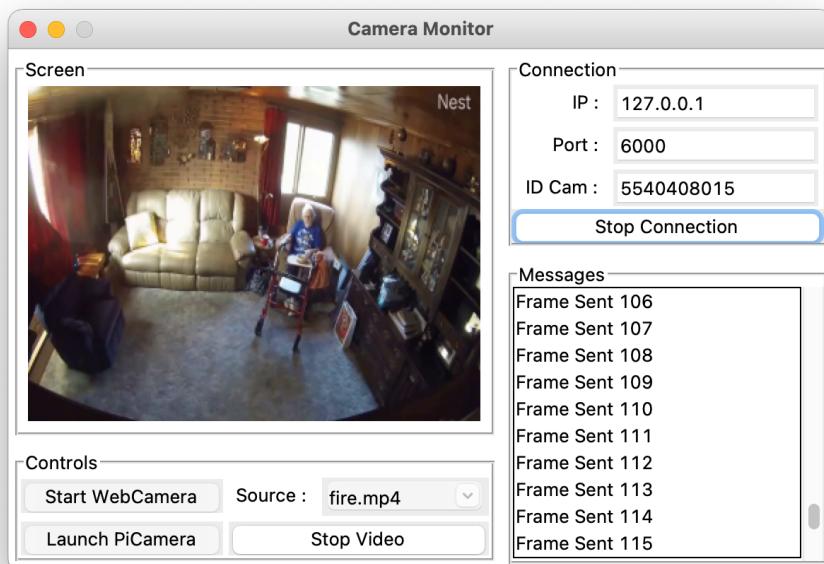


Figura 5.3: Interfaz gráfica del módulo de cámaras en envío de fotogramas.

Fuente : Elaboración Propia.

5.2. Módulo de servidor

Es el módulo central que gestiona distintos procesos como ser:

- Gestión de conexiones del módulo de cámaras.
- Recepción de fotogramas recibidos.
- Análisis de fotogramas según algoritmos de visión artificial.
- Notificación por correo electrónico.

5.2.1. Interfaz de consola

Durante la ejecución del servidor es necesario visualizar el estado del mismo. La opción inmediata y práctica es manejar mensajes del servidor por medio de la consola del sistema operativo. En la figura 5.4 se visualiza un ejemplo del estado inicial del servidor.



```
python main.py
> python main.py
2022-10-25 23:16:4 : ***** Welcome to the TCP-Server *****
2022-10-25 23:16:4 : Database created
2022-10-25 23:16:4 : Serving on : 127.0.0.1; on port : 6000
2022-10-25 23:16:4 : Listen connections :
```

Figura 5.4: Ejecución del servidor TCP.

Fuente : Elaboración propia

Además de registrar cada mensaje en pantalla se agrega la fecha y hora de su lanzamiento como se aprecia en la figura anterior.

5.2.2. Configuración y conexiones

Para la ejecución inicial del servidor es necesario ingresar las configuraciones relevantes para un funcionamiento consistente. Para tal objetivo se prevé de un archivo inicial de configuración en el cual se registran todos los valores en los que se basará el servidor. En la figura 5.5 se visualiza el archivo de configuración del servidor.

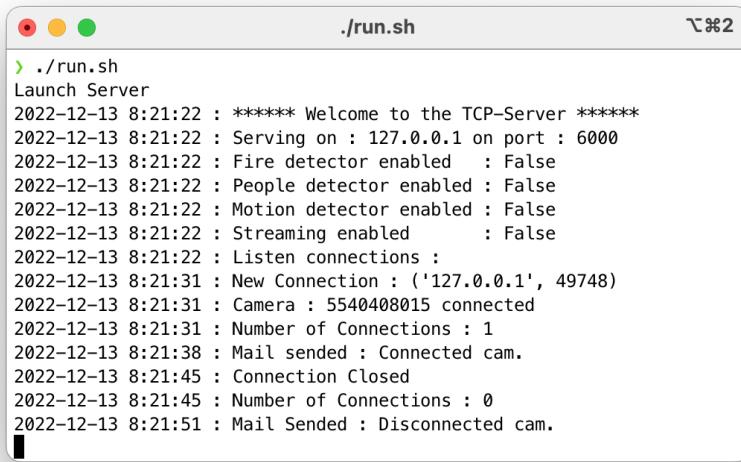
```
1 [server]
2 host=127.0.0.1
3 port=6000
4
5 [media]
6 folder_name=media
7
8 [email]
9 port=587
10 smtp_server=smtp.gmail.com
11 sender_mail=liveyesmart@gmail.com
12 receiver_mail=rodrigosergio93@gmail.com
```

Figura 5.5: Archivo de configuración del servidor.

Fuente : Elaboración propia

Este archivo permite definir información sobre el nombre del servidor, puerto de escucha y valores relevantes para la notificación por correo electrónico.

Las conexiones son gestionadas tambien por el servidor manteniendo informado al usuario sobre las nuevas conexiones por medio de la interfaz de consola como tambien por notificación que sera descrito más adelante. En la figura 5.6 se visualiza los diferentes mensajes referentes a las conexiones por parte de la interfaz de consola.



A screenshot of a terminal window titled ".run.sh". The window shows the output of a script named ./run.sh. The log starts with "Launch Server" and continues with various system messages and connection details. The text is as follows:

```
./run.sh
Launch Server
2022-12-13 8:21:22 : ***** Welcome to the TCP-Server *****
2022-12-13 8:21:22 : Serving on : 127.0.0.1 on port : 6000
2022-12-13 8:21:22 : Fire detector enabled : False
2022-12-13 8:21:22 : People detector enabled : False
2022-12-13 8:21:22 : Motion detector enabled : False
2022-12-13 8:21:22 : Streaming enabled : False
2022-12-13 8:21:22 : Listen connections :
2022-12-13 8:21:31 : New Connection : ('127.0.0.1', 49748)
2022-12-13 8:21:31 : Camera : 5540408015 connected
2022-12-13 8:21:31 : Number of Connections : 1
2022-12-13 8:21:38 : Mail sended : Connected cam.
2022-12-13 8:21:45 : Connection Closed
2022-12-13 8:21:45 : Number of Connections : 0
2022-12-13 8:21:51 : Mail Sended : Disconnected cam.
```

Figura 5.6: Mensajes de consola del servidor.

Fuente : Elaboración propia

5.2.3. Análisis de fotogramas

La característica central de este sistema de video-vigilancia es la detección de diferentes sucesos con el uso de algoritmos de visión por computadora e inteligencia artificial. A continuación se describen los algoritmos encargados para cada detección.

Detector de Fuego

Es el encargado de identificar la presencia de fuego en los fotogramas que recibe el servidor. La detección que realiza hace uso de detectores en “cascada”, que es un enfoque basado en Aprendizaje Automático en la que la función cascada se entrena a partir de muchas imágenes positivas y negativas, para luego utilizar este entrenamiento para realizar la detección en otro conjunto de imágenes.

La librería de visión por computadora “OpenCV” permite cargar los archivos .xml, que representan el entrenamiento, a su método HaarCascade, encargado de procesar la imagen. En la figura 5.7 se visualiza el código central de la detección de fuego, donde se aprecia el cargado del archivo de entrenamiento para el clasificador.

```

1 def detector(connection):
2     fire_cascade = cv2.CascadeClassifier(
3         'detectors/fire_detection.xml'
4     )
5     while connection.running:
6         frame, label = connection.get_frame(objetive='fire_detector')
7         if frame is not None:
8             fire = fire_cascade.detectMultiScale(
9                 image=frame,
10                scaleFactor=1.1,
11                minNeighbors=3,
12                flags=0
13            )
14            if len(fire) > 0:
15                connection.fire_detections.append((frame, label))

```

Figura 5.7: Código de detección de fuego.

Fuente : Elaboración propia

El clasificador realiza una detección multi-escala en base a los valores que se asignan al método en las líneas 10, 11 y 12.

Detector de silueta humana

Es el encargado de la detección de intrusos por la silueta humana, en base a los fotogramas recibidos por el servidor. Este detector hace uso del algoritmo HOG (Histograma de gradiente orientado) implementado por la librería “OpenCV”. Este método es entrenado para detectar gente que camina, que en la mayoría de los casos están de pie, así que no se espera que funcione correctamente en otros casos. La idea básica del método es la siguiente:

- La imagen se escanea con una ventana de detección de tamaño variable.
- Para cada posición y tamaño de la ventana de detección, esta se subdivide en celdas. Estas celdas son relativamente pequeñas: típicamente contienen solo una pequeña parte de la persona a detectar, tal vez el costado de un brazo o la parte superior de la cabeza.
- En cada celda, se calcula un gradiente para cada píxel y los gradientes se utilizan para llenar un histograma: el valor es el ángulo del gradiente y el peso es la magnitud del gradiente.
- Los histogramas de todas las celdas se juntan y se envían a un discriminador de aprendizaje automático para decidir si las celdas de la ventana de detección actual corresponden a una persona o no.

En la figura 5.8 se visualiza el código del detector de silueta huamana.

```

1 def detector(connection):
2     hog = cv2.HOGDescriptor()
3     hog.setSVMClassifier(
4         cv2.HOGDescriptor_getDefaultPeopleDetector()
5     )
6     while connection.running:
7         frame, label = connection.get_frame(objetive='people_detector')
8         if frame is not None:
9             gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
10            boxes, weights = hog.detectMultiScale(gray, winStride=(8,8) )
11            boxes = np.array(
12                [[x, y, x + w, y + h] for (x, y, w, h) in boxes]
13            )
14            if len(boxes) > 0:
15                print_log('i', f"People detected: {connection.cam_id}")
16                connection.people_detections.append((frame, label))

```

Figura 5.8: Código de detección de silueta humana.

Fuente : Elaboración propia

Detector de movimiento

Encargado de la detección de movimiento en base a los fotogramas recibidos por el servidor. El detector hace uso de métodos implementados en la librería “OpenCV”. En la figura 5.9 se detalla el código implementado que hace uso de métodos propios de la librería.

```

1 def detector(connection):
2     static_back = None
3     while connection.running:
4         frame, label = connection.get_frame(objetive='motion_detector')
5         if frame is not None:
6             gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
7             gray = cv2.GaussianBlur(gray, (21, 21), 0)
8             if static_back is None:
9                 static_back = gray
10                continue
11                diff_frame = cv2.absdiff(static_back, gray)
12                thresh_frame = cv2.threshold(
13                    diff_frame, 30, 255, cv2.THRESH_BINARY)[1]
14                thresh_frame = cv2.dilate(
15                    thresh_frame,
16                    None,
17                    iterations=3)
18                cnts, _ = cv2.findContours(
19                    thresh_frame.copy(),
20                    cv2.RETR_EXTERNAL,
21                    cv2.CHAIN_APPROX_SIMPLE)
22                if len(cnts) > 0:
23                    connection.motion_detections.append((frame, label))

```

Figura 5.9: Código de detección de movimiento.

Fuente : Elaboración propia

El método implementado convierte el fotograma a otro en escala de grises, aislando el color durante la detección. Define el primer fotograma como referencial para comparar con los siguientes fotogramas y determinar posibles diferencias que sugieren un movimiento.

5.2.4. Transmisión de video en vivo

Se implementa la clase que maneja la librería “VidGear” encargada de realizar la decodificación y codificación de video generando el archivo que indexa las porciones de archivos multimedia, generados por el proceso para su transmisión. En la figura 5.10 se detalla los valores de configuración necesaria para la correcta ejecución de la librería, especificando la resolución y el framerate del video a generar.

```
1 stream_params = {  
2     "-input_framerate": frame_rate,  
3     "-livestream": True,  
4     "-streams": [{  
5         "-resolution": "640x360",  
6         "-framerate": "30.0"}]  
7 }  
8 self.streamer = StreamGear(output = output_path, format = output_format,  
9                             **stream_params  
 )
```

Figura 5.10: Código de configuración de la librería VidGear.

Fuente : Elaboración propia

Capítulo 6

Pruebas

- 6.1. Pruebas de conexión al servidor**
- 6.2. Pruebas de notificación**
- 6.3. Prueba de transmisión en vivo**

Capítulo 7

Conclusiones

Concluimos que...

Capítulo 8

Ejemplos

8.1. Mas ejemplos de formato

8.1.1. Familia

typewriter (máquina de escribir)
sans serif
roman

8.1.2. Forma

texto en negritas
texto en itálicas
texto inclinado
texto en estilo máquina de escribir
TEXTO EN MAYÚSCULAS PEQUEÑAS

8.1.3. Tamaño

texto de prueba
texto de prueba

8.2. Listado

8.2.1. No numerados

- Item 1

- Item 2
- Item 3

8.2.2. Numerados

1. Item 1
2. Item 2
3. Item 3

8.3. Referenciación con APA

8.3.1. Citación como parte de párrafo

Un autor

Como menciona Apellido (2000), no es la única forma de citar.

Varios autores

Como menciona Apellido, Apellido, y Apellido (2001), no es la única forma de citar.

8.3.2. Citación en la parte final

Un autor

Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris citumoris totalis. (Apellido, 2000).

Varios autores

Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris citumoris totalis. (Apellido et al., 2001).

8.3.3. Citación con número de página

Como menciona Apellido (2000, p. 5), no es la única forma de citar.

Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris citumoris totalis. (Apellido et al., 2001, p. 7-12).

8.3.4. Citación anexos

Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat (Ver Anexo A).

8.4. Figuras

Referenciando a la figura 8.1.

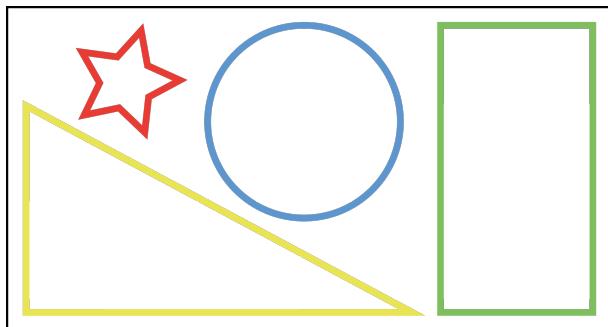


Figura 8.1: Explicación de la figura (Aqui)

Fuente: Adaptada de Apellido, N. (2000) *Nombre del libro*. Editorial o universidad que lo publicó.

8.5. Tablas

8.5.1. Corto

Referenciando a la tabla 8.4.

Tabla 8.1: Título de la tabla

| | Columna 1 | Columna 2 | Columna 3 |
|--------|-----------|-----------|-----------|
| Fila 1 | item | item | item |
| Fila 2 | item | item | item |
| Fila 3 | item | item | item |

Nota. Extraída de Apellido, N. (2000) *Nombre del libro*. Editorial o universidad que lo publicó.

8.5.2. Multipágina

Stique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Tabla 8.2.

Tabla 8.2: Titulo de tabla multipágina

Continua en la siguiente página.

Tabla 8.2 – Continuación de tabla previa

| | Columna 1 | Columna 2 | Columna 3 | Columna 4 |
|----------------|--|--|--|--|
| Fila 11 | Lorem ipsum dolor sit amet, consectetur adipiscing elit. | Lorem ipsum dolor sit amet, consectetur adipiscing elit. | Lorem ipsum dolor sit amet, consectetur adipiscing elit. | Lorem ipsum dolor sit amet, consectetur adipiscing elit. |
| Fila 12 | Lorem ipsum dolor sit amet, consectetur adipiscing elit. | Lorem ipsum dolor sit amet, consectetur adipiscing elit. | Lorem ipsum dolor sit amet, consectetur adipiscing elit. | Lorem ipsum dolor sit amet, consectetur adipiscing elit. |

Nota. Extraída de Apellido, N. (2000) *Nombre del libro*. Editorial o universidad que lo publicó.

8.6. Fórmulas matemáticas

Simple

$$e^{i\pi} + 1 = 0 \quad (8.1)$$

Matrices

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \quad (8.2)$$

Límites

$$\lim_{x \rightarrow \infty} \frac{3+x}{x^2} \quad (8.3)$$

8.7. Diagramas de flujo

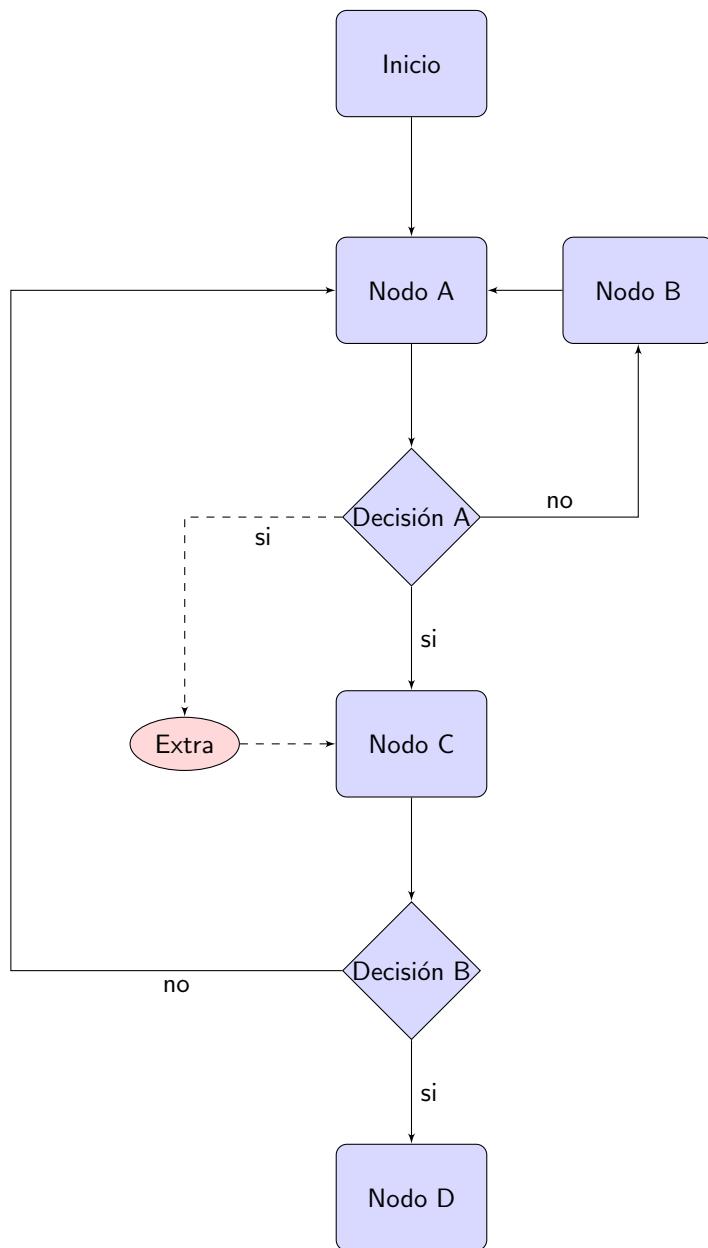


Tabla 8.3: Detalle de las pruebas realizadas

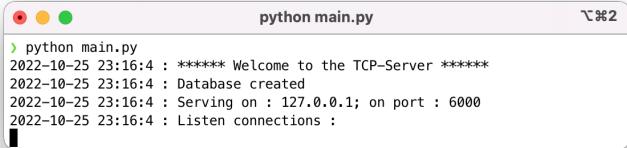
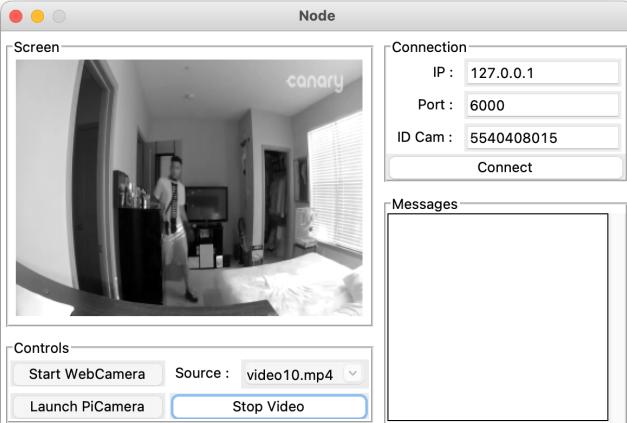
| Columna 1 | Columna 2 |
|--|---|
|  <pre> python main.py 2022-10-25 23:16:4 : ***** Welcome to the TCP-Server ***** 2022-10-25 23:16:4 : Database created 2022-10-25 23:16:4 : Serving on : 127.0.0.1; on port : 6000 2022-10-25 23:16:4 : Listen connections : </pre> | <ul style="list-style-type: none"> ■ Remote delivery ■ Immersive experiences ■ text proved |
|  | <ul style="list-style-type: none"> ■ Remote delivery ■ Immersive experiences ■ text proved |

Tabla 8.4: Detalle de las pruebas realizadas

| | Columna 1 | Columna 2 | Columna 3 |
|--------|------------------|------------------|------------------|
| Fila 1 | item | item | item |
| Fila 2 | item | item | item |
| Fila 3 | item | item | item |

Nota. Extraída de Apellido, N. (2000) *Nombre del libro*. Editorial o universidad que lo publicó.

| col1 | col2 | col3 |
|-----------------|-------------------------|-------------------------|
| Multiple row | cell2 cell5 cell8 | cell3 cell6 cell9 |

Referencias

- Apellido, N. (2000). *Título del libro* (Vol. 20). Editorial o universidad que lo publicó.
- Apellido, N., Apellido, N., y Apellido, N. (2001). *Título del libro*. Editorial o universidad que lo publicó.
- Austerberry, D. (2005). *The technology of video and audio streaming* (Vol. 1). ELSEVIER publisher.
- Bell, T. E., y Thayer, T. A. (1976). Software requirements: Are they really a problem? *IEEE Computer Society Press*, 61-68. Recuperado de https://static.aminer.org/pdf/PDF/000/361/405/software_requirements_are_they_really_a_problem.pdf
- Bootcamp-ia. (2019a, Noviembre). *Comprensión de las redes neuronales convolucionales 1d y 3d*. Recuperado de <https://programmerclick.com/article/89511747730/>
- Bootcamp-ia. (2019b, Noviembre). *Introducción a las redes neuronales convolucionales*. Recuperado de <https://bootcampai.medium.com/redes-neuronales-convolucionales-5e0ce960caf8>
- Briega, R. E. L. (2015, Septiembre). *Libro online iaar*. Recuperado de <https://iaarbook.github.io/>
- CCNA-Certification. (2022, Enero). *Http and https explained*. Recuperado de <https://study-ccna.com/http-https/>
- Chollet, F., y Allaire, J. J. (2018). *Deep learning with r*. Javvin Editiorals. Recuperado de https://bookdown.org/paul/computational_social_science/machine-learning-as-programming-paradigm.html#ref-Chollet2018-vf
- Gama, B., Sant'Ana, W., Lambert-Torres, G., Salomon, C., Bonaldi, E., da Silva, L. E., ... Steiner, F. (2021, 02). Fpga prototyping using the stemlab board with application on frequency response analysis of electric machinery. *IEEE Access, PP*, 1-1. doi: 10.1109/ACCESS.2021.3058059
- Ganis, M. (2010). Agile methods: Fact or fiction. *Research Gate*. Recuperado de <https://tcf.pages.tcnj.edu/files/2013/12/ganis-tcf2010.pdf>

- Garza, J. (2013, Septiembre). *Breve historia tcp/osi*. Recuperado de <https://www.javiergarzas.com/2013/09/tcpip-se-impuso-a-osi-2.html>
- Henry. (2022, Septiembre). *Python, ¿será el lenguaje más popular en 2023?* Recuperado de <https://blog.soyhenry.com/tag/mundo-henry/>
- Howard. (2022, Julio). *Redes cliente-servidor vs. redes peer-to-peer*. Recuperado de <https://community.fs.com/es/blog/client-server-vs-peer-to-peer-networks.html>
- Innovation, A. (2019, Octubre). *Qué son las redes neuronales y sus funciones*. Recuperado de <https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/>
- Lopez, R. (2016, Junio). *Tensorflow y redes neuronales*. Recuperado de <https://relopezbriega.github.io/blog/2016/06/05/tensorflow-y-redes-neuronales/>
- Lopez, R. (2017, Junio). *Introducción a la inteligencia artificial*. Recuperado de <https://relopezbriega.github.io/tag/inteligencia-artificial.html>
- Mancuso, G. (2021, Junio). *Modelo de desarrollo en cascada*. Recuperado de <https://blog.comparasoftware.com/modelo-de-desarrollo-en-cascada/>
- MarketsAndMarkets. (2020, Noviembre). *Video surveillance market with covid-19 impact analysis*. Recuperado de <https://www.marketsandmarkets.com/Market-Reports/video-surveillance-market-645.html>
- Norman, T. L. (2017). Chapter 6 - electronics elements: A detailed discussion originally from integrated security systems design. thomas norman: Butterworth-heinemann, 2015. updated by the editor, elsevier, 2016. , 95-137. Recuperado de <https://www.sciencedirect.com/science/article/pii/B9780128044629000063> doi: <https://doi.org/10.1016/B978-0-12-804462-9.00006-3>
- Oracle. (2015, Septiembre). *The java™ tutorials*. Recuperado de <https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html#:~:text=Definition%3A,address%20and%20a%20port%20number.>
- Ozer, J. (2017, Noviembre). *What is mpeg dash?* Recuperado de [https://www.streamingmedia.com/Articles/Editorial/What-Is-.../What-Is-HLS-\(HTTP-Live-Streaming\)-78221.aspx?utm_source=related_articles&utm_medium=gutenberg&utm_campaign=editors_selection](https://www.streamingmedia.com/Articles/Editorial/What-Is-.../What-Is-HLS-(HTTP-Live-Streaming)-78221.aspx?utm_source=related_articles&utm_medium=gutenberg&utm_campaign=editors_selection)
- Ozer, J. (2018, Diciembre). *What is hls (http live streaming)?* Recuperado de <https://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=79041>
- Royce, W. W. (1970). Managing the development of large software systems. *IEEE WESCON*, 328-338. Recuperado de <https://www.praxisframework.org/files/royce1970.pdf>

- Saha, S. (2018, Diciembre). *A comprehensive guide to convolutional neural networks*. Recuperado de <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- Solintel. (2018, Diciembre). *Sistemas de seguridad con cámaras ip*. Recuperado de <http://www.solintelsa.net/sistemas-de-videovigilancia/>
- Tagliaferri, L. (2017, Septiembre). An introduction to machine learning. *Digital Ocean*. Recuperado de <https://www.digitalocean.com/community/tutorials/an-introduction-to-machine-learning>
- Technologies Javvin, I. (2004). *Network protocols handbook*. Javvin Editi-
rals. Recuperado de <https://bkarak.wizhut.com/www/lectures/networks-07/NetworkProtocolsHandbook.pdf>
- Tejada, A. G. (2019). Aplicaciones de “deep learning” en entorno ros. *Research Ga-
te*. Recuperado de <https://idus.us.es/bitstream/handle/11441/102311/TFG-2951-GOMEZ%20TEJADA.pdf?sequence=1&isAllowed=y>
- Vijayanagar, K. R. (2021, Abril). *What is mpeg-dash video streaming protocol? how does mpeg-dash work?* Recuperado de <https://ottverse.com/mpeg-dash-video-streaming-the-complete-guide/>
- Wikipedia. (1994, Diciembre). *Modelo osi*. Recuperado de https://es.wikipedia.org/wiki/Modelo_OSI
- Wikipedia. (2012, Julio). *Perceptor multicapa*. Recuperado de https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa
- Wikipedia. (2020, Diciembre). *Videovigilancia ip*. Recuperado de https://es.wikipedia.org/wiki/Videovigilancia_IP
- Wikipedia. (2022, Octubre). *Cliente-servidor*. Recuperado de https://es.wikipedia.org/wiki/Videovigilancia_IP

Anexos

Anexo A: Manual de instalacion de la camara

Contenido de Anexo A

Anexo B: Instalación del servidor

Contenido de Anexo B

Anexo C: Instalación de la aplicación

Contenido de Anexo C