ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

PROTOTIPO DE UN SISTEMA WEB PARA EL CONTEO DE VEHÍCULOS EN TIEMPO REAL UTILIZANDO LA LIBRERÍA DE VISIÓN ARTIFICIAL OPENCV

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

EDGAR REINALDO GUERRA CARAPAS
reinaldo316@gmail.com

DIRECTOR: Ing DANNY SANTIAGO GUAMÁN LOACHAMÍN, MSc.
danny.guaman@epn.edu.ec

CODIRECTOR: Ing. LUIS ALBERTO MORALES ESCOBAR, MSc.
luis.moralesec@epn.edu.ec

Quito, Agosto 2016

DECLARACIÓN

Yo Edgar Reinaldo Guerra Carapas, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Edgar Reinaldo Guerra Carapas

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por Edgar Reinaldo Guerra Carapas bajo nuestra supervisión.

_____

Ing. DANNY GUAMÁN, MSc.
DIRECTOR DEL PROYECTO

_____

Ing. LUIS MORALES, MSc.
CODIRECTOR DEL PROYECTO

AGRADECIMIENTOS

A mis directores del proyecto de titulación, MSc. Danny Guamán y MSc Luis Morales, por el apoyo, paciencia y confianza brindada durante todo este tiempo.

A mis padres, Luis y Dilma por ser el pilar en el cual me apoyo física y moralmente.

A mis hermanos Luis y David por los ánimos recibidos.

A mis amigos por el apoyo y ayuda brindada, especialmente a Vinicio y Jessica.

A quienes conforman MARTEL Cía. Ltda. Por las facilidades prestadas al momento de utilizar sus activos, especialmente al Ing. Roberto Bonilla por su ayuda desinteresada y apoyo.

DEDICATORIA

A Dios por permitir culminar este proyecto, a mis padres que son las personas más importantes de mi vida y por las cuales estoy culminando este objetivo planteado.

Edgar

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGO

PRESENTACIÓN

De acuerdo a las estadísticas de la Secretaria de Movilidad de la Alcaldía Metropolitana de Quito, el parque automotor se incrementa aproximadamente en 30.000 vehículos por año. En tal virtud, en el Plan Maestro de movilidad 2009-2025 para la ciudad de Quito, se decidió instalar 1511 cámaras de video detección, de las cuales 185 corresponden a video cámaras que conforman el Sistema de Vigilancia de Transito (STV).

Por lo expuesto, el objetivo principal de este proyecto es diseñar e implementar un prototipo para conteo de vehículos, mediante un sistema web utilizando la librería de visión artificial de código abierto OPENCV y el lenguaje de programación Java. A través del procesamiento de las imágenes captadas se realizará el conteo de vehículos en las calles en donde se tengan instaladas las cámaras.

El prototipo consta de 4 módulos: adquisición y transmisión de video, recepción y procesamiento, almacenamiento y reportes.

El módulo de adquisición y transmisión de video está constituido por cámaras IP y se encarga de capturar y enviar el video por streaming a través de Internet hacia el módulo de recepción y procesamiento. El módulo de recepción y procesamiento se encarga de recibir el video transmitido por streaming desde las cámaras e implementar un algoritmo para el conteo de vehículos en tiempo real. El módulo de almacenamiento consta de una base de datos que almacena la información de las cámaras, usuarios del sistema, el reporte del conteo de automóviles en un intervalo de tiempo determinado, y la fecha y hora de conteo. Finalmente, el módulo de vistas y reportes contiene un conjunto de formularios que permiten realizar las operaciones CRUD (Create-Read-Update-Delete) de perfiles, usuarios y cámaras; además, permite mostrar la información del estado de flujo vehicular geolocalizada sobre un mapa y reportes históricos.

El prototipo es del tipo web, para así motivar a los usuarios a utilizar el sistema únicamente a través de un navegador, garantizando la compatibilidad del mismo con cualquier dispositivo conectado a Internet. El presente proyecto pretende ser la

base para el desarrollo de trabajos futuros que permitan determinar el flujo de tránsito vehicular, aprovechando la infraestructura de cámaras ya instaladas para el monitoreo, control y foto multas a cargo Empresa Pública Metropolitana de Movilidad y obras Públicas EPMMOP.

RESUMEN

El presente documento se ha organizado en 4 capítulos.

El capítulo 1 inicia con el estudio de los aspectos relevantes del fundamento teórico utilizado para el desarrollo del presente proyecto. Se revisa los conceptos básicos en torno a la visión artificial y reconocimiento de patrones, posteriormente se realizará el estudio de las tecnologías usadas en el proyecto; entre ellas se describe la librería para visión artificial de código abierto OPENCV y el framework Java Server Faces. Además, se realiza una revisión y estudio del estado del arte en torno a los sistemas de conteo vehicular a través del procesamiento de imágenes, con el objetivo de determinar los parámetros y criterios utilizados para realizar el conteo vehicular, el algoritmo empleado, arquitectura del sistema propuesto, y las recomendaciones en torno a trabajos futuros.

En el capítulo 2 se define los requerimientos funcionales y no funcionales del sistema de conteo de vehículos. A través de los diagramas UML se presenta el diseño funcional de cada uno de los módulos del sistema, además se incluye los aspectos más relevantes de la implementación del sistema.

En el capítulo 3 se realiza una descripción del escenario de pruebas empleado en la validación del sistema, y posteriormente se presenta los resultados de las pruebas de funcionalidad y de integración del sistema propuesto. Se describe los principales problemas encontrados en el desarrollo del algoritmo de conteo vehicular y las soluciones propuestas, que permitan reducir el error en el conteo vehicular.

Finalmente, en el capítulo 4 se presenta las conclusiones y las recomendaciones entorno al trabajo realizado, basado en los conocimientos adquiridos a lo largo del diseño e implementación del mismo.

CAPÍTULO 1: MARCO TEÓRICO

En este capítulo se incluye los conceptos básicos en torno a la visión artificial y reconocimiento de patrones. Se realizará el estudio de las tecnologías base a usarse en el proyecto, entre ellas se encuentra la librería para visión artificial de código abierto OPENCV que utiliza el lenguaje de programación Java y el framework Java Server Faces.

Posteriormente, se realizará una revisión y estudio del estado del arte en torno a los sistemas de conteo vehicular a través del procesamiento de imágenes. Para determinar los parámetros y criterios utilizados para realizar el conteo vehicular, el algoritmo empleado, arquitectura del sistema propuesto, y las recomendaciones en torno a trabajos futuros.

1.1 VISIÓN ARTIFICIAL

La visión artificial o visión por computador tiene como objetivo facilitar sistemas con capacidades de percepción humana para simular la apreciación del entorno y comprender los datos detectados, tomar las correcciones adecuadas e ilustrarse de esta experiencia con el fin de optimizar el rendimiento futuro de los mismos. Este campo ha tenido una gran evolución que va desde la aplicación de métodos de reconocimiento de patrones clásicos y procesamiento de imágenes hasta técnicas avanzadas en la comprensión de una imagen como la visión basada en modelos.

Para poder entender el concepto de visión artificial, se parte de la concepción de visión. La cual indica que es un proceso que por medio del sentido de la vista se recupera información del mundo exterior [1] [2]. Basándose en esta idea la visión artificial modela matemáticamente el proceso de la percepción visual de los seres vivos y permite simular estas capacidades visuales a través de una computadora [3]. En la actualidad se percibe este tipo de recursos en diferentes campos como por ejemplo la que brinda la empresa israelí Mobileye que desarrolla desde fines de los 90 cámaras especiales de alta fidelidad para sistemas basados en visión artificial, las que proporcionar un grado de autonomía a los vehículos en donde se encuentran instaladas, y son utilizadas para prevenir accidentes de tránsito.

Existen muchos campos en los cuales se ha incursionado con la visión artificial, entre ellos vehículos, manufactura, interpretación de imágenes aéreas, etc.

En la actualidad existe muchas formas de configurar un sistema de visión artificial pero la más básica e imprescindible consta de estos dos elementos que se especifican en la Tabla 1.1.

Tabla 1.1 Configuración sistema de visión artificial

Elemento	Descripción
Sensor óptico	Para poder captar la imagen, este puede ser desde una cámara de video, cámara de fotografía, scanner, etc
Computador	Para crear y ejecutar los algoritmos de visión artificial además de almacenar las imágenes y/o videos a utilizarse

1.1.1 FASES PARA LA IMPLEMENTACIÓN DE VISIÓN ARTIFICIAL [4]

El proceso que se realiza para implementar la visión artificial se puede subdividir en cuatro áreas principales, las cuales se detallan en la Tabla 1.2.

Tabla 1.2 Procesos de la visión artificial

Área	Descripción
Adquisición y representación	Procedimiento para obtención de una imagen.
Procesamiento de filtrado	Procedimiento en el cual se emplean métodos de reducción de ruido y modificación de los detalles de una imagen
Segmentación	Procedimiento de fraccionamiento en objetos de interés a una imagen de muestra
Descripción, reconocimiento de patrones e Interpretación	Procedimiento para tratamiento de particularidades útiles para distinguir un tipo de objeto de otro, identificación de objetos, búsqueda de significado a los mismos

1.1.1.1 Adquisición y representación

Una imagen digital se encuentra compuesta por un conjunto de pixeles, los cuales ofrecen información de una región de la imagen. Esta información puedes ser el brillo, la profundidad de color o el modelo de color que utiliza, como por ejemplo el modelo RGB (rojo, verde, azul).

Para obtener una imagen digital se distinguen dos pasos:

- **Captura. –** Para que se produzca la captura se recurre al uso de dispositivos, generalmente del tipo óptico, con el que se obtiene la información de una escena. Entre estos dispositivos se tiene por ejemplo las cámaras de video, cámaras fotográficas, escáner, etc.
- **Digitalización. –** Es el proceso de transformación de una imagen analógica a digital, en el cual se distinguen dos pasos significativos, el muestreo y la cuantificación.

1.1.1.2 Procesamiento de filtrado

En este procedimiento se aplican operaciones y transformaciones a las imágenes digitales como paso previo a la segmentación y reconocimiento, con el objetivo de destacar elementos seleccionados de estas. Las imágenes y sonidos son los ejemplos de señales más comunes, en la Tabla 1.3 se encuentra un listado con los principales tipos y operaciones respectivas.

Tabla 1.3 Operaciones o transformaciones en visión artificial.

Tipo / agrupación	Operaciones / transformaciones
Operaciones sobre pixeles	Operaciones aritmético - lógicas
	Operaciones geométricas
Operaciones en el histograma	Aumento y reducción de contraste
	Ecualizado del histograma
Filtrado espacial	Filtros de suavizado
	Filtros de obtención de contornos
	Filtro Laplaciano
Dominio de la frecuencia	Transformada de Fourier
	Teorema de convolución
Operaciones morfológicas	Filtros morfológicos
	Operaciones morfológicas

Todas las operaciones que se pueden realizar en este procedimiento se puede interpretar como filtros, esto sería como un mecanismo de transformación de una señal que entra a la cual se le aplica una función para obtener otra señal a la salida.

1.1.1.3 Segmentación

La segmentación es un proceso en el cual a partir de una imagen, se genera otra en la que cada pixel perteneciente a esta tendrá una etiqueta asociada que lo distinga, con esto se obtiene objetos agrupados formados por pixeles con la misma etiqueta.

Existen algunos tipos de segmentación entre los que se destacan:

- Segmentación basada en umbralización
- Segmentación basada en detección de contornos
- Segmentación basada en crecimiento de regiones

En aplicaciones reales se suele usar una combinación de los distintos métodos listados anteriormente para ajustarse al problema para el cual se busque una solución.

1.1.1.4 Descripción, reconocimiento de patrones e interpretación

En esta fase se usan varios algoritmos que pueden distinguir objetos de un conjunto llamado universo de trabajo.

Como primer paso para el reconocimiento de objetos se debe obtener las características discriminantes o rasgos. Dichos rasgos se obtienen convirtiendo un objeto del universo de trabajo en un vector, cuyo valor se conoce como patrón del objeto y cuyas componentes se llaman características discriminantes. Este universo se encuentra dividido en clases, que a su vez está constituido de varios objetos.

La clasificación de un objeto empieza con la determinación de su patrón, esta determinación es un proceso que requiere usar características del mismo objeto que se deben obtener mediante la utilización de algún procedimiento algorítmico.

Después de esto se aplican funciones discriminantes que permiten determinar el grado de pertenencia del patrón a cada una de las clases.

Estas funciones discriminantes se calculan con una muestra de aprendizaje, la misma es un conjunto de patrones similares a los que se desea determinar, si la muestra es lo suficientemente grande a partir de ella también se crea un conjunto de prueba que debe ser independiente es decir no deben tener elementos en común. Cuando se determina la pertenencia del patrón a una clase se dice que el clasificador ha sido construido. En la Figura 1.1 se puede encontrar un gráfico referencial.

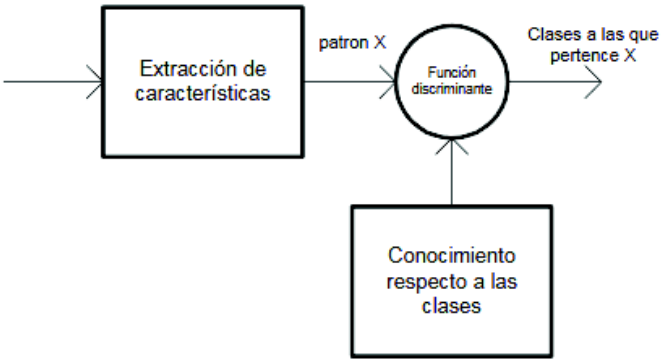


Figura 1.1 Esquema general de un clasificador [4]

1.2 TECNOLOGÍAS EMPLEADAS EN EL PROYECTO

1.2.1 LIBRERÍA DE VISIÓN ARTIFICIAL OPENCV [6]

OPENCV es un conjunto de librerías de código abierto para visión artificial bajo licencia BSD[1], tanto para uso académico y comercial. Esta escrito y optimizado en C/C++, cuenta con interfaces para lenguajes de programación como C++, C, Python y Java, es compatible con numerosos sistemas operativos como Windows, Linux, Mac OS, iOS y Android.

[1] BSD (Berkeley Software Distribution). Es una licencia de software libre permisiva, la cual permite la redistribución, uso y modificación del software.

Figura 1.2 Línea del tiempo de OPENCV [6]

En la Figura 1.2 se puede apreciar la línea de tiempo con respecto a las versiones liberadas, en el presente proyecto se empleó la versión 3.0 que se encuentra disponible en [7].

1.2.1.1 Estructura de OPENCV

OPENCV mantiene una estructura modular la cual se muestra en la Figura 1.3 y como se observa en la Tabla 1.4 se plantea una descripción básica de cada módulo perteneciente a la librería OpenCV:

Tabla 1.4 Descripción de los módulos de OPENCV

Módulo	Característica
Core	Modulo en el cual se define las estructuras de datos y funciones básicas utilizadas por el resto de módulos
Imgproc	Módulo de procesamiento de imágenes como: filtrado, transformaciones lineales, conversión de espacios de color, etc
Video	Módulo de análisis de video que incluye estimación y seguimiento de objetos, entre otros
Calib3d	Módulo con múltiples algoritmos geométricos para vista múltiple básica, calibración de cámara, estimación de objetos, reconstrucción 3D, etc
Features2d	Módulo que contiene descriptores, comparadores y detectores de características sobresalientes
Objdetect	Módulo para detección de objetos predefinidos como caras, vehículos, etc
Highgui	Moduló para captura de video, códec de video y capacidades de entrada y salida de datos
Gpu	Módulo que contiene algoritmos acelerados de los distintos módulos de OPENCV

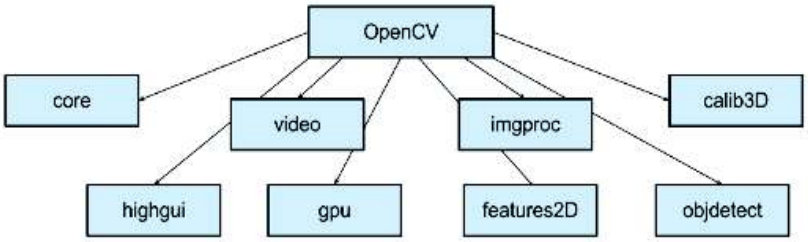


Figura 1.3 Estructura modular de la librería OPENCV [6]

1.2.1.2 OPENCV CON SOPORTE PARA EL LENGUAJE DE PROGRAMACIÓN JAVA

OPENCV en febrero del 2013 oficialmente dio a conocer la integración con el lenguaje de programación Java, permitiendo llevar a cabo las operaciones disponibles en la librería, como por ejemplo la búsqueda de bordes, líneas, círculos, etc. Con este lenguaje de programación. Todo lo relacionado a Java y OPENCV se encuentra debidamente documentado y se encuentra disponible en [8].

1.2.2 JAVA [9]

Una de las principales características de Java es la de disponer de un gran número de librerías y/o paquetes de clases para la realización de aplicaciones, todas están organizadas en tres grandes grupos llamadas ediciones Java, las cuales se mencionan en la Tabla 1.5.

Tabla 1.5 Ediciones de Java

Nombre de edición.	Descripción.
Java Standard Edition (Java SE)	En este grupo se encuentran las clases de uso general, es decir todas aquellas que se utilizan en cualquier tipo de aplicación, tales como: cadenas, colecciones, etc. Además se incluye las clases para creación de entornos gráficos, applets y demás
Java Enterprise Edition (Java EE)	Esta colección proporciona las clases necesarias para el desarrollo de aplicaciones empresariales del tipo multicapa incluye Java SE y muchas más extensiones
Java Micro Edition (Java ME)	Edición enfocada en los dispositivos móviles, línea blanca y/o dispositivos electrónicos que contiene clases y especificaciones que posibilitan la creación de aplicaciones móviles en entorno Java, es una versión ajustada de Java SE que se enfoca en las necesidades específicas de estos tipos de productos

1.2.2.1 Java Enterprise Edition [10]

El modelo de aplicaciones de Java EE específica una arquitectura para la implementación de servicios en múltiples niveles, ofreciendo accesibilidad, escalabilidad y la capacidad de gestionar las aplicaciones de nivel empresarial.

1.2.2.1.1 Componentes de Java EE

En la Figura 1.4 se puede observar los componentes de Java EE.

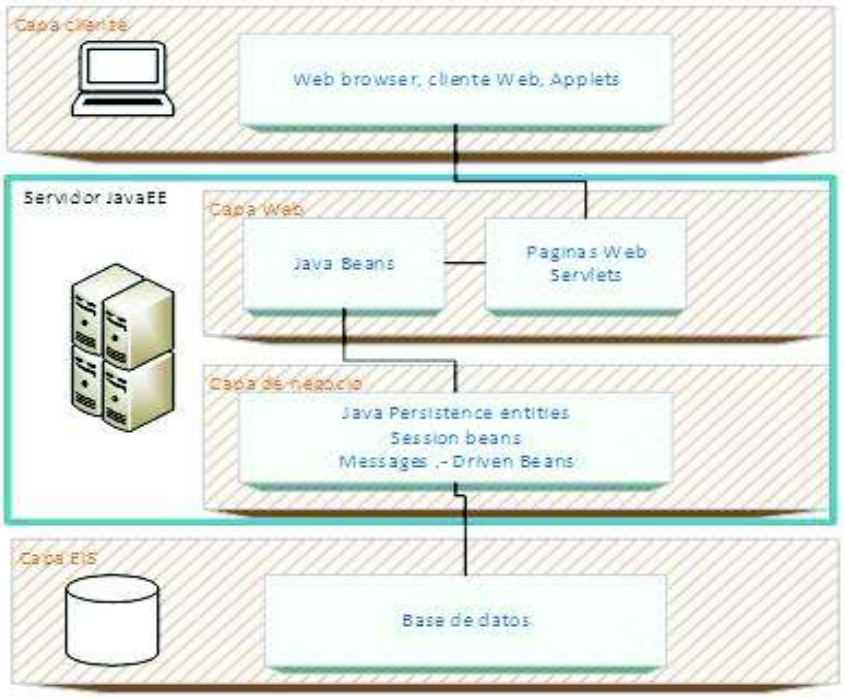


Figura 1.4 Componentes Java EE [10]

Java EE hace uso de un modelo de n niveles, con esto la lógica de la aplicación se divide en componentes de acuerdo a su función. Un componente de Java EE es una unidad utilitaria de software, la cual es totalmente independiente y forma parte de una aplicación Java EE con sus respectivas clases y archivos relacionados. En Java EE generalmente se consideran aplicaciones de tres capas debido a que están distribuidos en tres lugares diferentes: la maquina cliente conocida como la capa cliente, el servidor Java EE como tal, que cuenta con la capa web y la capa de negocio y el servidor de base de datos que se encuentra en la capa EIS

(Enterprise Information Server), como se especifica en la figura 1.4. La especificación Java EE define los siguientes componentes:

- Componentes de la capa cliente como aplicaciones y applets
- Componentes de la capa Web como: Servlets, Páginas JSP, Java Server Faces basado en servlets y JSP que se ejecutan en un servidor.
- Componentes de la capa de negocio como los EJB (Enterprise Java Beans) que se ejecutan el servidor.

1.2.3 JAVA SERVER FACES [11]

JSF es un framework del tipo MVC (Modelo-Vista-Controlador) que se basa en el API de Servlets, la cual facilita características de plantillas y creación de componentes compuestos. El API suministra elementos en forma de etiquetas definidas en páginas XHTML mediante el framework Facelets.

La tecnología de Java Server Faces permite una disgregación clara del comportamiento y presentación de aplicaciones Web, esto permite a los desarrolladores de un equipo enfocarse en una sola parte del desarrollo y proporcionar un modelo sencillo para unir las partes.

1.2.3.1 Características

JSF tiene las siguientes características:

- Creación de interfaces por medio de vistas con componentes gráficos
- Unión entre las interfaces con los datos de la aplicación mediante los beans gestionados.
- Conversión de datos y validación automática de la entrada del usuario.
- Navegación entre vistas.
- A partir de la especificación 2.0 un modelo estándar de comunicación Ajax entre la vista y el servidor
- Soporte para los servidores de aplicaciones e IDE’s como: Eclipse, GlassFish, etc

- Una fácil integración con los distintos frameworks entre la capa de negocio y la capa de persistencia: Spring, JPA, etc.

1.2.3.2 Implementaciones JSF

La tecnología JSF es una estructura de componentes del lado del servidor que permite crear aplicaciones web basadas en Java. Por lo cual existen múltiples implementaciones de referencia de esta tecnología. Las implementaciones más usadas se muestran en la Tabla 1.6.

Tabla 1.6 Implementaciones de JSF

Implementación	Autor	Descripción
Mojarra	Sun	Es una de las implementaciones más populares, y se encuentra incluida en los servidores de aplicaciones Java Enterprise como GlassFish y JBoss
MyFaces	Fundación Apache	Desarrollada por la comunidad Apache, está incluida en el servidor de aplicaciones Gerónimo

1.3 REVISIÓN DEL ESTADO DEL ARTE

Como punto de partida se realiza un estudio del estado del arte en torno a los sistemas de conteo vehicular mediante el procesamiento de imágenes. Para ello, se estudió tres trabajos similares que mantienen una estrecha relación con el presente proyecto. El propósito de esta revisión es determinar los parámetros y criterios utilizados para realizar el conteo vehicular, el algoritmo empleado, arquitectura del sistema propuesto, y las recomendaciones en torno a trabajos futuros. Esta información será utilizada como insumo para la fase de análisis de requerimientos y diseño del sistema prototipo, los documentos relacionados con los proyectos analizados se encuentran en [12] [13] y [14].

1.3.1 SISTEMA DE MONITORIZACIÓN Y CONTROL DE TRÁFICO EN CARRETERA [12]

El trabajo en mención [12], tiene como principal objetivo desarrollar una aplicación capaz de detectar y clasificar vehículos, además de realizar un seguimiento a partir del procesamiento de video de secuencias de tráfico diurnas.

1.3.1.1 Consideraciones del proyecto

Las consideraciones empleadas en este proyecto se describen en las siguientes secciones.

1.3.1.1.1 Cámara

En la elección de la cámara se optó por el modelo pin-hole. La misma que permite calcular las coordenadas de un objeto en la imagen en función de las coordenadas en el mundo real, por lo cual se definió los siguientes parámetros intrínsecos y extrínsecos para la cámara:

- Parámetros intrínsecos de la cámara:
  - Foco: FU = 400, FV = -700.
  - pixeles: u0 = -160, v0= -120.
- Parámetros extrínsecos de la cámara:
  - posición horizontal = -6 m.
  - Altura =-5 m.
  - ángulo de elevación.α = -10.2 grados.

1.3.1.1.2 Tamaño de vehículos

El tamaño de los vehículos es otro parámetro destacable ya que en el proyecto tiene como uno de sus objetivos la identificación de los mismos. Por ello, establecen las medidas promedio de vehículos pequeños y grandes.

En la Tabla 1.7 se puede apreciar las dimensiones establecidas para la medida de los vehículos**:**

Tabla 1.7 Dimensiones de vehículos [12]

Vehículo	**∆X**	**∆Y**	**∆Z**
Pequeños	1.6 m	0.8 m	3 m
Grandes	2.5 m	1.6 m	7m

Con estos parámetros y con el uso de un algoritmo desarrollado con la librería OPENCV se ejecuta a la detección con tracking y posterior clasificación de los vehículos con una proyección 3d con líneas de colores:

- Azul si el carro se clasifica como vehículo grande.
- Verde si el carro se clasifica como vehículo pequeño.

Este sistema distinguirá entre vehículos grandes y pequeños, aunque también se indica que podrá clasificar vehículos como motos, pero esa opción no está depurada.

1.3.1.2 Algoritmo propuesto

El algoritmo propuesto es el que se puede apreciar en la Figura 1.5.

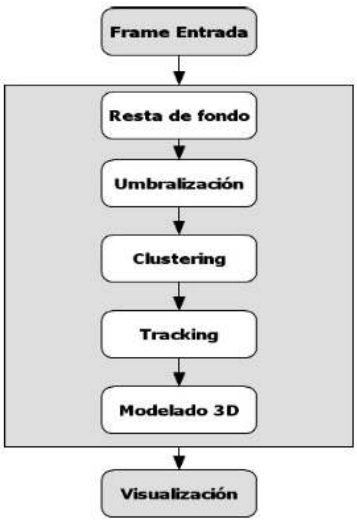


Figura 1.5 Algoritmo para detección [12]

- **Frame de entrada. –** En esta sección se utilizan videos pregrabados en formato de video AVI y en escala de grises.
- **Resta de fondo. –** Basado en la idea de que el fondo permanece constante, se puede estimar los vehículos en movimiento restando a la imagen el fondo que permanece inmóvil.

- **Umbralización. –** Después del proceso de resta de fondo los pixeles clasificados como fondo presentan un valor de gris igual a 0 y los pixeles de los vehículos en movimiento están entre 1 y 255.
  La umbralización consiste en dar un valor de 255 (1 lógico) a todos los pixeles pertenecientes al primer plano y de 0 (0 lógico) a los pixeles pertenecientes al fondo, con lo cual solo se tendrá blanco y negro.
- **Operadores morfológicos. –** Se utilizan con imágenes binarias para reducir efectos de ruido y mejorar la calidad.
- **Clustering. –** Consiste en agrupar los pixeles en un solo grupo, el algoritmo tiene dos procesos el de clustering y el de limpieza del clustering.
- **Tracking. –** Es el seguimiento a los centroides de cada uno de los clusters de la etapa anterior.
- **Modelado 3D. –** Alrededor de los vehículos detectados se dibuje una proyección 3D para presentar los datos, para esto se necesita calibrar la cámara (aspectos intrínsecos y extrínsecos).
- **Visualización. –** Se muestra el resultado obtenido después de la ejecución del algoritmo.

1.3.1.3 Arquitectura propuesta

Este proyecto consta de un solo modulo cuyo funcionamiento consiste en capturar el video a partir de grabaciones de video hechas con anterioridad, esto se procesa con el algoritmo propuesto y se muestra al usuario.
Cabe recalcar que la arquitectura propuesta no está diseñada para ser ejecutada en tiempo real.

1.3.1.4 Características relevantes

Aquí se recoge las conclusiones y recomendaciones que pueden contribuir con el desarrollo del proyecto que se está proponiendo:

- Debido a la existencia de alteración meteorológicas se puede obtener clusters no deseados, dichos clusters generan falsos positivos en lo concerniente a la detección.

- Se tiene problemas por no realizar correctamente la liberación del tracking de un vehículo cuando este sale fuera de los límites de la ventana.
- Existen problemas cuando hay que distinguir dos vehículos que se encuentran muy cerca.
- El algoritmo puede tener problemas cuando circulan camiones en paralelo debido a su elevado tamaño

1.3.2 CONTEO AUTOMÁTICO DE VEHÍCULOS [13]

El trabajo en mención [13] tiene como objetivo el proponer una técnica para encontrar el número de vehículos en un video tomado en un segmento de vía típico con condiciones de iluminación aceptables

1.3.2.1 Consideraciones del proyecto

Las consideraciones empleadas en este proyecto para la detección y conteo vehicular son los siguientes:

1.3.2.1.1 Muestras de video

Se utilizaron tomas de un segmento típico de vía. Entiéndase por este un tramo unidireccional de una vía. Se evitó tener trayectos en los cuales el flujo de vehículos detalle más de una dirección.

1.3.2.1.2 Cámara

Para la elección de la cámara se definieron los siguientes parámetros intrínsecos y extrínsecos:

- Parámetros intrínsecos de la cámara:
  - Frames por segundo: 25 o 30.
  - Codecs: MJPEG, Xvid, DivX.
- Parámetros extrínsecos de la cámara:
  - Desplazamiento en y: el video se toma a alturas variables.

- Matriz de la cámara: se hará con respecto al eje horizontal en un rango de -45 ± 30 grados.
- Iluminación: todos los videos utilizados serán durante el día.

1.3.2.1.3 Áreas de interés

Son áreas pre definidas, en las cuales se ejecuta el algoritmo propuesto para reducción de procesamiento.

1.3.2.1.4 Conteo de vehículos

El criterio utilizado para el conteo es trazar una línea perpendicular a la calle y contar cuando el centro de masa del vehículo haya cruzado dicha línea. Además, se marca el este centro para no volverlo a contar.

1.3.2.2 Algoritmo propuesto

El algoritmo propuesto en este proyecto es el que se puede apreciar en la Figura 1.6.

En el inicio del algoritmo se solicita la selección de un video o cámara. A continuación, se selecciona una región de interés y una perspectiva la para no procesar regiones que no hagan parte de la vía.

El siguiente paso es la estimación previa de fondo la cual tiene como algoritmo base la estimación de fondo y posterior reconocimiento del primer plano con lo que detecta los objetos en la escena. Sobre esto se aplica una máscara de bordes de fondo que es un operador cany para unir líneas de contorno del vehículo.

Inmediatamente se realiza la estimación de fondo por FDGStatmodel y con este algoritmo separa cuadros en los que hay movimiento de los que no lo hay para luego realizar una variación de perspectiva.

Una vez realizado esto se realiza el seguimiento de bloques y la aplicación del algoritmo de reconocimiento, seguimiento, clasificación y conteo.

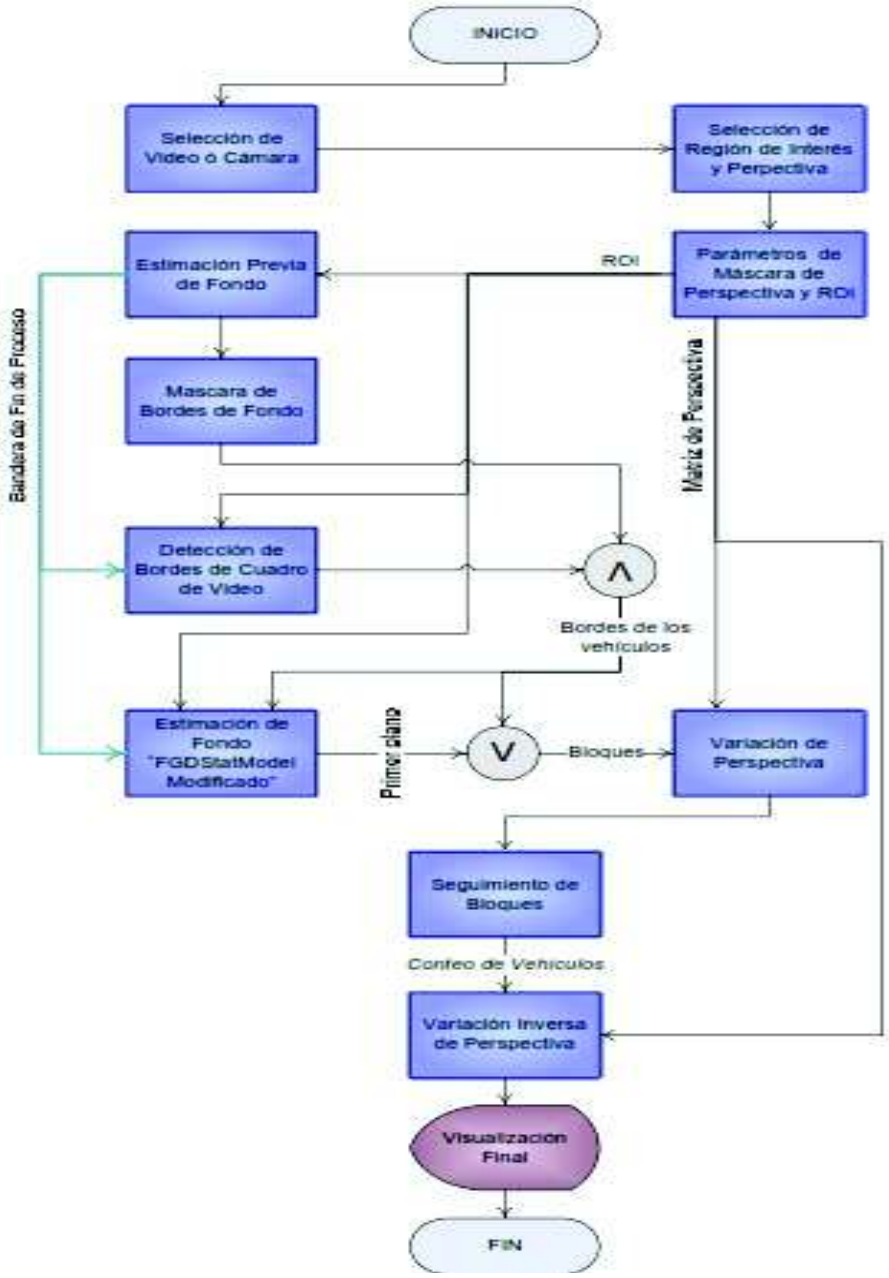


Figura 1.6 Algoritmo para detección y conteo [13]

1.3.2.3 Arquitectura propuesta

Este proyecto consta de un solo modulo cuyo funcionamiento consiste en capturar el video a partir de grabaciones hechas con anterioridad. El video se procesa con el algoritmo propuesto y se muestra al usuario. De igual forma, cabe recalcar que el algoritmo y su arquitectura no fueron diseñados para ser utilizados en tiempo real.

1.3.2.4 Características Relevantes

- Se obtiene errores pequeños en el conteo, cuando la ubicación de la cámara se encuentra a 90 grados con respecto a la vía. De esta manera, se evita la unión de vehículos por perspectiva. Además, su ubicación debe ser a una altura lo suficientemente alta para que los vehículos largos sean capturados en el cuadro de la escena.

1.3.3 ALGORITMO PARA CONTEO VEHÍCULAR EN TIEMPO REAL CON BASE EN FRANJAS DE INTERÉS [14]

El trabajo en mención [14], tiene como objetivo principal el realizar un conteo vehicular en tiempo real, procesando únicamente franjas de interés. Adicionalmente, se calcula la velocidad y densidad de tráfico.

1.3.3.1 Consideraciones del proyecto

Las consideraciones empleadas en este proyecto para la detección y conteo vehicular en tiempo real son las siguientes:

1.3.3.1.1 Captura de video

A continuación se lista las condiciones, que deben cumplir los videos a procesarse en el algoritmo:

- Luminosidad estándar.
- Ángulo de la cámara favorable
- Nitidez óptima.
- Resolución buena

1.3.3.1.2 Franja de interés y línea de conteo

Para optimizar el procesamiento y conteo el algoritmo solo se ejecuta en determinadas franjas de interés, lo cual permite al usuario definir la dirección del flujo vehicular.

Además, para el conteo se tiene en cuenta que el objeto cumpla con los requisitos de área, ancho y largo para ser considerado como válido.

1.3.3.2 Algoritmo propuesto

En la Figura 1.7 se encuentra el algoritmo propuesto en el proyecto en mención.

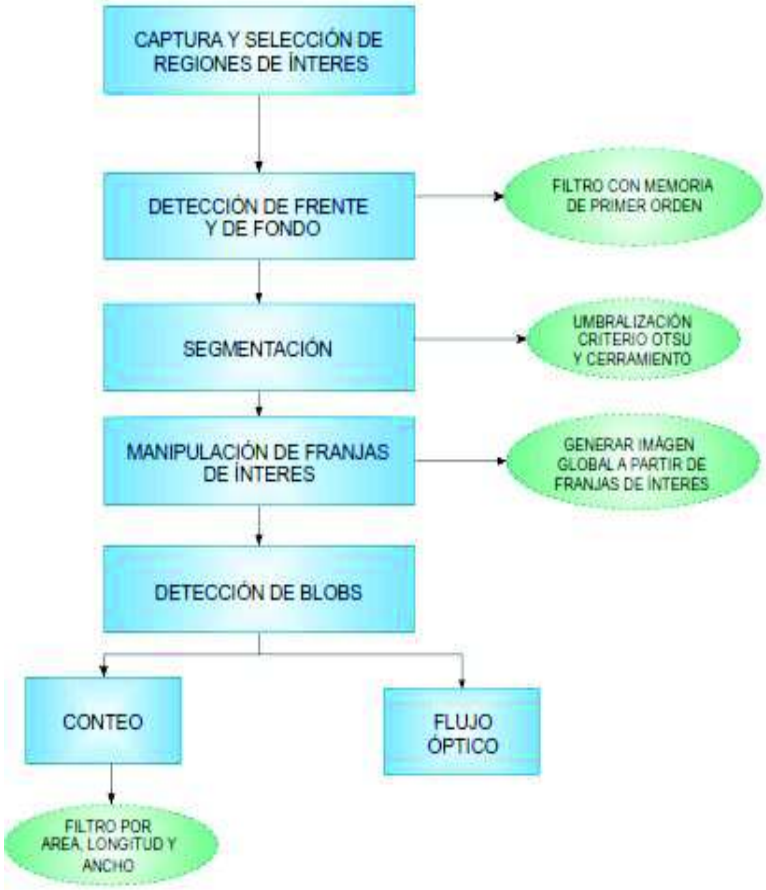


Figura 1.7 Algoritmo para detección y conteo en tiempo real [14]

- **Captura y selección de regiones de interés. –** Módulo que permite al usuario seleccionar las áreas de interés y línea de conteo. El sistema solo procesa las franjas escogidas.
- **Detección de frente y fondo. –** Módulo que diferencia la imagen de primer plano y el fondo, desarrollado un filtro con memoria de primer orden.
- **Segmentación. –** Esta parte realiza dos procedimientos que son los siguientes:

    - **Umbralización. –** Para obtener una imagen binaria de los objetos presentes en la franja de interés (método OTSU)
    - **Cerramiento. –** Operadores morfológicos de cerramiento para mejorar la conectividad de los objetos en la imagen binaria
- **Manipulación de franjas. –** A partir de la formación de la imagen se obtiene otra imagen por franja, que representa el tránsito vehicular y a partir de estas se realiza el conto y cálculo de los parámetros de tráfico.
- **Conteo. –** Las regiones detectadas continúan moviéndose. Cuando el final de esta región pasa por la línea de conteo se incrementa el contador en uno, siempre y cuando se cumpla con los requisitos de largo, ancho y área
- **Flujo óptico. –** Se realiza el cálculo del centro y se aplica el algoritmo de flujo óptico.

1.3.3.3 Arquitectura propuesta

Este proyecto consta de un solo módulo, cuyo funcionamiento consiste en capturar el video a partir de grabaciones hechas con anterioridad y ofrecer la posibilidad de realizar el procesamiento en tiempo real.

1.3.3.4 Características Relevantes

- El algoritmo se realizó buscando la optimización del tiempo de ejecución, para la simplificación de las ejecuciones de procesos.
- El tiempo de procesamiento se puede reducir aún más generando un hilo por cada región procesada y aprovechar mejor el procesador.

CAPÍTULO 2: DISEÑO E IMPLEMENTACIÓN

En este capítulo se definirá los requerimientos funcionales y no funcionales del sistema de conteo de vehículos, para ello se emplearán casos de uso. A través de los diagramas UML necesarios se presentará el diseño funcional de cada uno de los módulos del sistema.

Se incluirá los aspectos más relevantes de la implementación del sistema, que incluye el algoritmo para el conteo de vehículos en tiempo real utilizando la librería OPENCV y el uso del framework Java Server Faces para la implementación de los formularios que permitan realizar las operaciones CRUD y reportes.

2.1 METODOLOGÍA DE DESARROLLO

La metodología a utilizar para el proceso de desarrollo del proyecto es Scrum. Esta es una metodología ágil que se caracteriza por ser iterativa e incremental. El objetivo de la metodología es maximizar el retorno de la inversión (ROI).

Se basa en levantar primero la funcionalidad de mayor valor para el cliente y en los principios de inspección continua, adaptación, auto-gestión e innovación [15] [16].

2.1.1 CARACTERÍSTICAS DE SCRUM

Las principales características de la metodología son las siguientes:

- Equipos auto dirigidos
- Utiliza reglas para crear un entorno ágil de administración de proyectos
- No describe prácticas de ingeniería
- Los requerimientos se listan como ítems
- El producto se construye con Sprints que tienen un tiempo de duración
- Desarrollo de software iterativos incrementales basados en prácticas agiles
- Enfatiza valores y prácticas de gestión, sin pronunciarse sobre requerimientos, prácticas de desarrollo, implementación y demás cuestiones técnicas.

2.1.2 PROCESO DE DESARROLLO DE SCRUM [17]

En Scrum todo el trabajo que se realiza es listado en el Product Backlog, que es una lista de todos los requerimientos en torno al producto, obtenida después del análisis de las respectivas historias de usuario.

El proyecto es desarrollado durante una serie de iteraciones temporales llamadas Sprints. Al comienzo de cada Sprint debe existir una Sprint Planning Meeting que es una reunión durante la cual se da prioridad a los ítems que se encuentran en la lista Product Backlog. Esta tarea la realiza el Product Owner que es el encargado del proyecto, inmediatamente el Scrum Team, que es el equipo de trabajo, seleccionará las tareas que serán completadas durante el Sprint. Estas tareas son llevadas a una nueva lista llamada Sprint Backlog que es la lista que contiene las tareas que el equipo se compromete a realizar durante el Sprint. Durante el Sprint el equipo mantiene Daily Meetings, que son reuniones diarias donde se analiza los avances, problemas y retrasos de una manera objetiva.

Al final del respectivo Sprint, en una reunión llamada Sprint Review Meeting, se debe mostrar la funcionalidad completa planificada. Un esquema de lo explicado anteriormente se encuentra en la Figura 2.1.

Figura 2.1 Esquema general de Scrum [17]

2.2 REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

2.2.1 PROPUESTA DEL PROTOTIPO

El presente proyecto propone un prototipo de sistema web para el conteo de vehículos en tiempo real, utilizando la librería de visión artificial OPENCV que incluirá fundamentalmente las siguientes funcionalidades:

- Captura de video por medio de dos cámaras, las mismas se enviarán hacia un servidor, a través de la red por un servicio de streaming perteneciente a la cámara.
- Procesamiento en tiempo real del streaming recibido por un servidor, para realizar el conteo vehicular, mediante un algoritmo desarrollado con la librería OPENCV.
- Almacenamiento del conteo vehicular, datos de usuarios e información de las cámaras (localización, parámetros de configuración, etc.) en una base de datos relacional.
- Visualización de formularios web que permitan realizar las operaciones CRUD de usuarios, parámetros iniciales de configuración del sistema y cámaras. Además de reportes de conteo vehicular, los cuales se mostrarán geolocalizados sobre un mapa y en gráficos de barras.

Teniendo en cuenta las funcionalidades antes mencionadas el diagrama modular propuesto del proyecto se muestra en Figura 2.2.

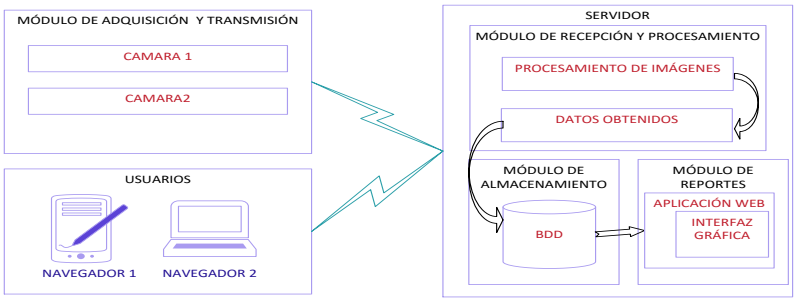


Figura 2.2 Diagrama modular del sistema propuesto

2.2.2 CASOS DE USO

Se utilizó casos de uso como un medio para posteriormente definir los requerimientos funcionales del prototipo.

2.2.2.1 Actores del sistema

Los actores definen un rol que cumplirá un usuario del sistema. Para el proyecto se definieron los siguientes actores:

- Principales:
    - Administrador. – Es quien manejará todas las funcionalidades que proporcione el sistema.
    - Operador de cámaras. – Se encargará de la administración de las cámaras y sus parámetros de configuración.
    - Usuario. – Es el que solicitará reportes de conteo en la aplicación
- De apoyo:
    - Cámaras. – Encargadas de proporcionar el stream de video en tiempo real

2.2.2.2 Diagrama de casos de uso

El diagrama que se muestra en la Figura 2.3 expresa la relación que tiene cada actor con los diferentes usos que tendrá el sistema. A partir del análisis de este diagrama se realizará las historias de usuario del sistema.

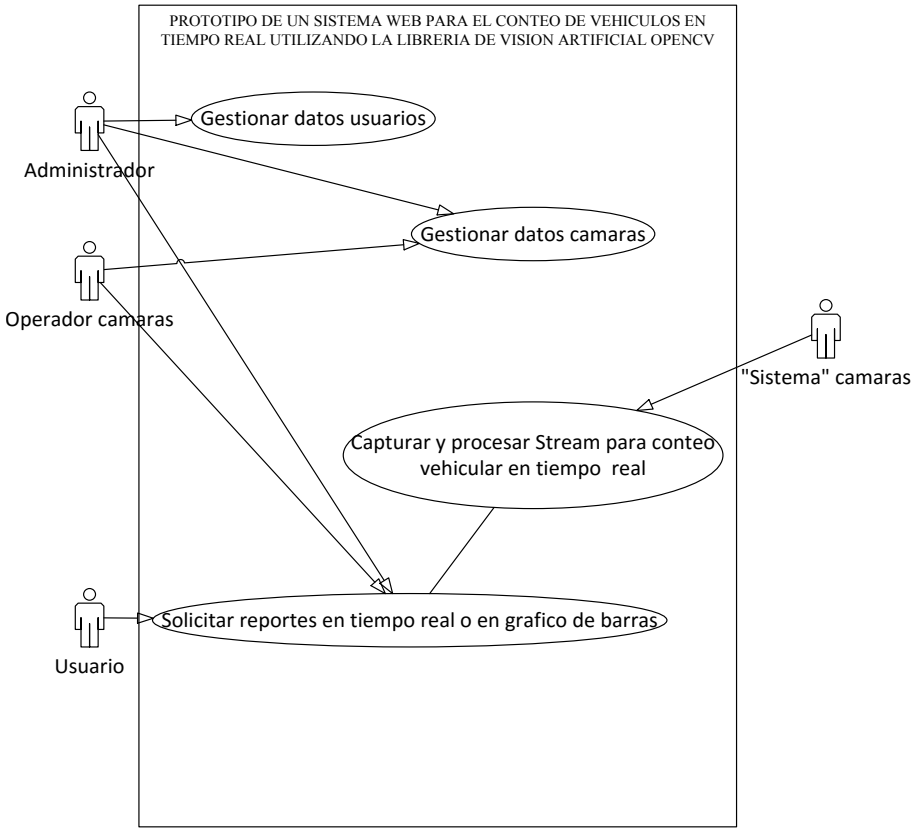


Figura 2.3 Diagrama de casos de uso general del sistema

2.2.3 HISTORIAS DE USUARIO Y CRITERIOS DE ACEPTACIÓN

A continuación se presenta las historias de usuario y los criterios de aceptación para cada uno de los escenarios identificados. En la Tabla 2.1, se puede apreciar el formato de la historia de usuario a emplear.

Tabla 2.1 Formato historia de usuario Scrum [18]

Actividad		Enunciado de la Historia de usuario		
Identificador (ID) de la Historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#) de Escenario

- **Actividad. –** Expresa la relación entre la historia de usuario con la actividad relacionada al rol, de acuerdo al diagrama de casos de uso.
- **Identificador (ID) de la historia. –** Código de identificación de la historia de usuario.
- **Rol. –** Es el actor que realiza la funcionalidad que se está describiendo.
- **Característica / Funcionalidad. –** Constituye la funcionalidad que el rol quiere o necesita hacer en el sistema.
- **Razón / Resultado. –** Describe la razón por la que se ejecuta la acción.
- **Número (#) de Escenario. –** Es el número que identifica a los escenarios asociados a la historia. Es importante mencionar que una historia de usuario puede tener muchos escenarios.

Por cada escenario identificado se han definido un conjunto de criterios de aceptación. Dichos criterios de aceptación le permitirán al cliente/usuario validar que se cumple con la funcionalidad inicialmente definida. En la Tabla 2.2, se puede observar el formato sugerido para los criterios de aceptación:

Tabla 2.2 Formato criterio de aceptación Scrum [18].

Criterios de Aceptación				
Criterio de Aceptación	Contexto	Datos entrada	Evento	Resultado / Comportamiento esperado

- **Criterio de Aceptación (Título). –** Describe el comportamiento esperado del escenario.
- **Contexto. –** Proporciona una descripción de las condiciones que generan el escenario.
- **Datos de entrada. –** describe los parámetros de entrada para realizar la acción requerida.
- **Evento:** Representa la acción que el usuario ejecuta, para cumplir con el contexto.
- **Resultado / Comportamiento esperado:** Representa el comportamiento esperado del sistema al generarse el evento.

A continuación se describen las historias de usuario y los criterios de aceptación planteados en este proyecto, de acuerdo al formato antes descrito. Cabe indicar que las historias de usuario se han organizado de acuerdo a los actores definidos en el sistema.

2.2.3.1 Rol de Administrador

Para el rol Administrador se definió 5 historias de usuario y cada una de estas consta de 5 escenarios con su respectivo criterio de aceptación. A manera de ejemplo la primera historia de usuario se mostrará completa, para el resto se mostrará únicamente las historias de usuario, si se desea ver los respectivos criterios de aceptación referirse al Anexo B.1. En la Tabla 2.3 y en la Tabla 2.4, se presenta la historia de usuario de la actividad *Gestionar Usuarios* con sus respectivos criterios de aceptación

Tabla 2.3 Historia de usuario *Gestionar Usuarios*

Gestionar datos usuario		Gestionar Usuario		
Identificador (ID) Historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#) de escenario
HUA001	Como un administrador…	…necesito gestionar usuarios en el sistema…	…con la finalidad de crear, actualizar y eliminar usuarios dentro del sistema.	5

Tabla 2.4 Criterios de aceptación de la HU *Gestionar Usuarios*

Criterios de aceptación de HUA001				
Criterio de Aceptación	Contexto	Entradas de datos	Evento	Resultado / Comportamiento esperado
Se crea un usuario nuevo…	…en caso que de ingresar correctamente todos los campos solicitados…	…nombre, apellido, username, password, e-mail y fecha de nacimiento…	…cuando el administrador genera la solicitud de creación de usuario…	…el sistema realiza la correcta creación del nuevo usuario en la base de datos y se mostrará en una tabla resumen.
No se crea un nuevo usuario	…en caso de no ingresar correctamente todos los campos solicitados o ya exista un registro igual…	Ninguna	…cuando el usuario administrador genera la solicitud de creación de usuario…	…el sistema no realiza la creación del nuevo usuario en la base de datos y retorna un mensaje de error.
Se actualiza un usuario	…en caso que de ingresar correctamente todos los campos solicitados…	Opcional: Nombre, Apellido, Username, password, e-mail, fecha de nacimiento	…cuando el usuario administrador genera la solicitud de actualización de usuario…	…el sistema realiza la correcta actualización del usuario en la base de datos y mostrará los datos en la tabla resumen
No se actualiza un usuario	…en caso de no ingresar ninguna información…	Ninguna	…cuando el usuario administrador genera la solicitud de actualización de usuario…	…el sistema no realiza la actualización del usuario en la base de datos y retorna un mensaje de error.
Se elimina un usuario	…en caso de seleccionar correctamente el usuario a eliminar…	Ninguna	…cuando el usuario administrador genera la solicitud de eliminación de usuario…	…el sistema realiza la correcta actualización del estado del usuario eliminado en la base de datos.

En la Tabla 2.5, se presenta el resto de las historias de usuario referente a la actividad de *Gestionar datos usuario* perteneciente al rol administrador.

Tabla 2.5 Historia de usuario de la actividad *Gestionar datos usuario*

Identificador (ID)Historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número de escenarios
HUA002	Como un administrador…	…necesito gestionar los roles de usuario…	…con la finalidad de crear, actualizar y eliminar roles.	5
HUA003	Como un administrador….	….necesito gestionar los menús de usuario…	…con la finalidad de crear, actualizar y/o eliminar menús de la vista web.	5
HUA004	Como un administrador…	…necesito gestionar la relación entre los roles de usuario y los menús…	…con la finalidad de asignar los menús a los que tendrá acceso determinado rol.	5
HUA005	Como un administrador…	…necesito gestionar la relación entre los roles de usuario y los usuarios…	…con la finalidad de asignar un rol a los usuarios que tendrá acceso al sistema	1

2.2.3.2 Rol de Operador de cámaras

Para el rol operador de cámaras se definió 2 historias de usuario y una de estas consta de 5 criterios de aceptación y la otra de 2. A manera de ejemplo la primera historia de usuario se mostrará completa, para el resto se mostrará únicamente las historias de usuario, si se desea ver los respectivos criterios de aceptación referirse al Anexo B.2. En la Tabla 2.6 y en la Tabla 2.7, se presentan la historia de usuario de gestionar cámaras y sus respectivos criterios de aceptación.

Tabla 2.6 Historia de usuario *Gestionar cámaras*

Gestionar datos cámara		Gestionar cámaras		
Identificador (ID) Historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#) de escenario
HUO001	Como un operador de cámaras…	…necesito gestionar las cámaras en el sistema…	…con la finalidad de crear, actualizar o eliminar sus datos.	1

Tabla 2.7 Criterios de aceptación de la HU *gestionar cámaras*

Criterios de aceptación de HUO001				
Criterio de Aceptación (Título)	Contexto	Entradas de datos	Evento	Resultado / Comportamiento esperado
Se crea una cámara nueva…	…en caso de ingresar correctamente todos los campos solicitados…	Nombre, Marca, modelo, coordenadas de latitud, coordenadas de longitud, url, localización	…cuando el usuario administrador u operador generan la solicitud de creación de una cámara…	…el sistema realiza la correcta creación de una cámara en la base de datos y se mostrará en una tabla resumen.
No se crea una cámara nueva…	…en caso de no ingresar correctamente todos los campos solicitados o ya exista un registro igual…	Nombre, Marca, modelo, coordenadas de latitud, coordenadas de longitud, url, localización	…cuando el usuario administrador u operador generan la solicitud de creación de una cámara…	…el sistema no realiza la creación de una cámara en la base de datos y retorna un mensaje de error.
Se actualiza una cámara…	…en caso de ingresar correctamente todos los campos solicitados…	Nombre, Marca, modelo, coordenadas de latitud, coordenadas de longitud, url, localización	…cuando el usuario administrador u operador generan la solicitud de actualización de una cámara…	…el sistema realiza la correcta actualización de una cámara en la base de datos y mostrará los datos en la tabla resumen
No se actualiza una cámara…	…en caso de no ingresar ninguna información…	Nombre, Marca, modelo, coordenadas de latitud, coordenadas de longitud, url, localización	…cuando el usuario administrador u operador generan la solicitud de actualización de una cámara…	…el sistema no realiza la actualización de una cámara en la base de datos y retornará un mensaje de error.
Se elimina un usuario…	…en caso de seleccionar correctamente el usuario a eliminar…	Ninguna	…cuando el usuario administrador u operador generan la solicitud de eliminación de una cámara…	…el sistema realiza la correcta actualización del estado de una cámara eliminada en la base de datos y ya no se mostrará los datos en la tabla resumen.

En la Tabla 2.8 se presenta la restante historia de usuario referente a la actividad de *Gestionar datos cámaras* pertenecientes al rol de operador de cámaras.

Tabla 2.8 Historias de usuario de la actividad *Gestionar datos cámara*

Gestionar datos cámara		Modificar parámetros cámara		
Identificador (ID) Historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número de Escenario
HUO002	Como un operador de cámaras…	…necesito modificar los parámetros de tráfico en el sistema…	…con la finalidad de actualizar los parámetros de tráfico en el sistema utilizados para generar reportes.	1
HUO003	Como un operador de cámaras…	…necesito capturar el stream de video proveniente de las cámaras....	…con la finalidad de procesar el stream de video en tiempo real con el algoritmo propuesto.	2
HUO004	Como un operador de cámaras…	…necesito procesar el stream de video en tiempo real…	…con la finalidad de obtener el dato de conteo vehicular de los mismos	2
HUO005	Como un operador de cámaras…	…necesito almacenar el conteo vehicular, fecha y hora de la toma de muestra…	…con la finalidad de disponer de estos datos cuando se necesite presentarlos	2

2.2.3.3 Rol usuario

Para el rol usuario se definió 3 historias de usuario y dos de estas consta de 2 criterios de aceptación y la última consta con cuatro. A manera de ejemplo la primera historia de usuario se mostrará completa, para el resto se mostrará únicamente las historias de usuario, si se desea ver los respectivos criterios de aceptación referirse al Anexo B.3.

En la Tabla 2.9 y en la Tabla 2.11, se presentan la historia de usuario *gestionar cámaras* y sus respectivos criterios de aceptación.

Tabla 2.9 Historia de usuario *Visualizar cámara*

Solicitar reporte en tiempo real		Modificar visualizar cámara		
Identificador (ID) Historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#)Escenario
HUU001	Como un usuario…	…necesito visualizar la ubicación de las cámaras en un mapa...	…con la finalidad de obtener los datos de conteo que esta está realizando en una ubicación determinada.	2

En la Tabla 2.10, se presenta el resto de las historias de usuario referente a la actividad de *Solicitar reporte en tiempo real* perteneciente al rol de usuario.

Tabla 2.10 Historias de usuario de la actividad *Solicitar reporte en tiempo real*

Identificador (ID) de la Historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#) de Escenario
HUU002	Como un usuario…	…necesito visualizar la ubicación de las cámaras en un mapa...	…con la finalidad de obtener los datos de conteo que esta está realizando en una ubicación determinada.	2
HUU003	Como un usuario…	…necesito visualizar la ubicación de las cámaras en un mapa...	…con la finalidad de obtener los datos de conteo que esta está realizando en una ubicación determinada.	2
HUU004	Como un usuario…	…necesito visualizar la ubicación de las cámaras en un mapa...	…con la finalidad de obtener los datos de conteo que esta está realizando en una ubicación determinada.	2
HUU005	Como un usuario…	…necesito visualizar la ubicación de las cámaras en un mapa...	…con la finalidad de obtener los datos de conteo que esta está realizando en una ubicación determinada.	2

Tabla 2.11 Criterios de aceptación de la HU *Visualización cámara*

Criterios de aceptación de HUU001					
Criterio de Aceptación (Título)	Contexto	Entradas de datos	Salidas de datos	Evento	Resultado / Comportamiento esperado
Se visualiza una cámara…	…en caso que el usuario ingrese al aplicativo…	Ninguna	Nombre cámara, localización, coordenada latitud, coordenada longitud	…cuando el sistema realiza correctamente la consulta a la base de datos…	…el sistema permitirá la correcta visualización de una cámara, la cual dispondrá de los datos de conteo de la misma y se mostrara su ubicación sobre un mapa.
No se visualiza una cámara…	…en caso que la aplicación no esté funcionando...	Ninguna	Ninguna	…cuando el servidor o las cámaras estén fuera de servicio o no se disponga del servicio de internet…	…el sistema generará un mensaje de error, en caso de no poder ingresar en el mismo

2.2.4 ANÁLISIS DE REQUERIMIENTOS

En base a la funcionalidad inicial definida en el plan del proyecto de titulación, las historias de usuario y sus criterios de aceptación, se definió los requisitos funcionales y no funcionales del proyecto.

2.2.4.1 Requisitos Funcionales

En la Tabla 2.12**¡Error! No se encuentra el origen de la referencia.** y la Tabla .13 se menciona los requisitos funcionales del sistema.

Tabla 2.12 (1) Requisitos funcionales del proyecto

Requisitos funcionales	
1	El sistema permitirá al administrador, gestionar usuarios con la finalidad de crear, actualizar y/o eliminar los datos de los mismos en el sistema
2	El sistema permitirá al administrador, gestionar los roles de usuarios, con la finalidad de crear, actualizar y/o eliminar los datos de los mismos en el sistema
3	El sistema permitirá al administrador, gestionar los menús de usuarios, con la finalidad de crear, actualizar y/o eliminar los datos de los mismos en el sistema
4	El sistema permitirá al administrador, gestionar la relación entre los roles de usuario y los menús , con la finalidad de asignar los menús a los que tendrá acceso determinado rol de usuario en el sistema
5	El sistema permitirá al administrador, gestionar la relación entre los roles de usuario y los usuarios, con la finalidad de asignar un rol a los usuarios que tendrá acceso al sistema
6	El sistema permitirá al operador, gestionar las cámaras, con la finalidad de crear, actualizar y/o eliminar los datos de las mismos en el sistema
7	El sistema permitirá al operador, modificar los parámetros de tráfico en el sistema., con la finalidad de actualizar los parámetros de tráfico en el sistema utilizados para generar reportes
8	El sistema permitirá al usuario normal, visualizar la ubicación de las cámaras en un mapa, con la finalidad de obtener los datos de conteo que esta está realizando y el estimado vehicular
9	El sistema permitirá al usuario normal, generar reportes de conteo vehicular, con la finalidad de obtener gráficos estadísticos del conteo vehicular de determinada cámara en el tiempo
10	El sistema permitirá al usuario normal, crear una cuenta de usuario nuevo , con la finalidad de ingresar al sistema y utilizar sus servicios
11	El sistema permitirá al usuario normal, actualizar mis datos y mi pastor de acceso , con la finalidad de corregir posibles fallos en los datos personales

Tabla 2.13 (2) Requisitos funcionales del proyecto

12	El sistema deberá capturar el stream de video proveniente de las cámaras, con la finalidad de procesar el stream de video en tiempo real con el algoritmo propuesto.
13	El sistema deberá procesar el stream de video en tiempo real, con la finalidad de obtener el dato de conteo vehicular de los mismos
14	El sistema deberá almacenar el conteo vehicular, fecha y hora de la toma de muestra, con la finalidad de generar búsqueda de estos datos y ser presentados
15	El sistema deberá permitir a los usuarios acceder al sistema especificando su rol de usuario, con la finalidad de proveer a cada usuario un menú de acceso diferente dependiendo de su rol en el sistema
16	El sistema deberá cargar en tablas los datos de los usuarios, cámaras, roles, relaciones, etc. En vistas diferentes, con la finalidad de proveer a los usuarios dependiendo de su rol de acceso, tablas informativas de los usuarios, cámaras, roles, relaciones, etc.

2.2.4.2 Requisitos no funcionales

En la Tabla 2.14 se encontrarán lo requisitos no funcionales del sistema

Tabla 2.14 Requisitos no funcionales del proyecto

Requisitos no funcionales	
1	Utilizar la librería de código abierto para visión artificial OPENCV
2	Utilizar el lenguaje de programación JAVA en conjunto con la librería OPENCV
3	Desarrollar la aplicación web mediante el framework JSF
4	Emplear el API de Google Maps para reportes de conteo vehicular

2.2.5 PLANIFICACIÓN SCRUM

Como parte de la planificación de Scrum se definió la Product backlog list y el número de Sprints en los que se desarrollará el sistema. Posteriormente, se priorizó

y se obtuvo las respectivas Sprint backlog list, que serán las listas de los requerimientos a solventar durante cada Sprint.

2.2.5.1 Planificación del Sprint

Esta planificación consiste en el refinamiento de la Product backlog list con el objetivo de entrar en detalle de lo que se va a desarrollar. Como primera parte se define la duración y la cantidad de Sprints estimadas para el desarrollo del proyecto. Tomando en cuenta los tiempos estimados de desarrollo de proyectos similares se estableció que por lo menos se debe tener 4 Sprints con una duración de un mes [19]. Con esta directriz se calcula las horas efectivas de trabajo que se dedicará al desarrollo del proyecto. El cálculo contempla una dedicación de 8 horas diarias de trabajo de lunes a viernes, con esto se obtiene los valores expuestos en la Tabla 2.15.

Tabla 2.15 Horas disponibles para desarrollo

Numeración	Actividad	Horas
1	Duración de un sprint	160
2	Actividades dedicadas a reuniones diarias	4
3	Actividades de investigación	15
4	Otras Actividades (permisos, movilización, etc.)	15
5	Total horas actividades (2+3+4)	34
6	Horas disponibles para desarrollo (1-5)	126
7	Horas efectivas de desarrollo (80 % de 6)	102

Las horas efectivas de desarrollo corresponden al 80 por ciento del total de horas disponibles para el desarrollo [20]. Esta consideración se hace debido a que

durante el Sprint pueden surgir actividades ajenas al desarrollo y que podrían poner en riesgo el cumplimiento de lo planificado. Una vez hecho esto se realiza el refinamiento en tareas de cada característica que este listada en la respectiva Sprint Backlog list.

2.2.5.2 Product backlog list

Una vez desarrollado las historias de usuarios se lista las características del producto en la Product backlog list con el siguiente formato expuesto en la Tabla 2.16.

Tabla 2.16 Formato de la Product backlog list [21]

Identificador (ID) de la Historia	Enunciado de la Historia	Alias	Estado	Dimensión / Esfuerzo	Sprint	Comentarios

- **Identificador (ID) de la Historia. –** Es el código que identifica la historia de usuario.
- **Enunciado de la Historia. –** Es el enunciado de la historia.
- **Alias. –** Es una descripción de la historia de usuario que sirve para identificarla más fácilmente.
- **Estado. –** Identifica los estados de la historia que podría tener durante el ciclo de vida, estos pueden ser: vacío, planificada, en proceso, hecho y descartada.
- **Dimensión / Esfuerzo. –** Es la medida del esfuerzo que tomaría al equipo de desarrollo atender la historia de usuario, en el caso del presente proyecto se utilizó puntos de historia los cuales dan una medida de complejidad a cada una, a mas alto el valor mayor dificultad.
- **Iteración (Sprint). –** Identifica al Sprint en el cual será desarrollada esta historia de usuario, esta asignación dependerá del nivel de prioridad que se asigne a cada historia.
- **Comentarios** Especifica comentarios o detalles de las historias descritas. En la Tabla 2.17, Tabla 2.18 y Tabla 2.19 se encuentra la Product Backlog list del prototipo propuesto.

Tabla 2.17 (1) Product backlog list

Identificador (ID) de la Historia	Enunciado de la Historia	Alias	Estado	Dimensión / Esfuerzo	Sprint	Comentarios
HUA001	Como un administrador, necesito gestionar usuarios, con la finalidad de crear, actualizar y/o eliminar los datos de los mismos en el sistema	Gestionar usuarios	Concluido	15	sprint 2	esta historia de usuario contempla 5 criterios de aceptación
HUA002	Como un administrador, necesito gestionar los roles de usuarios, con la finalidad de crear, actualizar y/o eliminar los datos de los mismos en el sistema	Gestionar roles	Concluido	3	sprint 2	esta historia de usuario contempla 5 criterios de aceptación
HUA003	Como un administrador, necesito gestionar los menús de usuarios, con la finalidad de crear, actualizar y/o eliminar los datos de los mismos en el sistema	Gestionar menús	Concluido	6	sprint 3	esta historia de usuario contempla 5 criterios de aceptación
HUA004	Como un administrador, necesito gestionar la relación entre los roles de usuario y los menús , con la finalidad de asignar los menús a los que tendrá acceso determinado rol de usuario en el sistema	Gestionar rol-menú	Concluido	5	sprint 3	esta historia de usuario contempla 5 criterios de aceptación
HUA005	Como un administrador, necesito gestionar la relación entre los roles de usuario y los usuarios, con la finalidad de asignar un rol a los usuarios que tendrá acceso al sistema	Gestionar usuario-rol	Concluido	5	sprint 3	esta historia de usuario contempla 5 criterios de aceptación

Tabla 2.18 (2) Product backlog list

Identificador (ID) de la Historia	Enunciado de la Historia	Alias	Estado	Dimensión / Esfuerzo	Sprint	Comentarios
HUO001	Como un operador, necesito gestionar cámaras, con la finalidad de crear, actualizar y/o eliminar los datos de las mismos en el sistema	Gestionar cámaras	Concluido	6	sprint 4	esta historia de usuario contempla 5 criterios de aceptación
HUO002	Como un operador, necesito modificar los parámetros de tráfico en el sistema., con la finalidad de actualizar los parámetros de tráfico en el sistema utilizados para generar reportes	Actualizar parámetros trafico	Concluido	1	sprint 4	esta historia de usuario contempla 2 criterios de aceptación
HUU001	Como un usuario, necesito visualizar la ubicación de las cámaras en un mapa, con la finalidad de obtener los datos de conteo que esta está realizando y el estimado vehicular	visualizar conteo	Concluido	20	sprint 1	esta historia de usuario contempla 2 criterios de aceptación
HUU002	Como un usuario, necesito generar reportes de conteo vehicular, con la finalidad de obtener gráficos estadísticos del conteo vehicular de determinada cámara en el tiempo	generar reportes	Concluido	23	sprint 3	esta historia de usuario contempla 2 criterios de aceptación
HUU003	Como un usuario, necesito gestionar una cuenta de usuario, con la finalidad crear y actualizar los datos	cuenta usuario	Concluido	2	Sprint 4	esta historia de usuario contempla 4 criterios de aceptación

Tabla 2.19 (3) Product backlog list

Identificador (ID) de la Historia	Enunciado de la Historia	Alias	Estado	Dimensión / Esfuerzo	Sprint	Comentarios
HUO003	Como un operador, necesito capturar el stream de video proveniente de las cámaras, con la finalidad capturar el stream de video proveniente de las cámaras	Capturar stream	Concluido	4	sprint 1	esta historia de usuario contempla 2criterios de aceptación
HUO004	Como un operador, necesito procesar el stream de video en tiempo real, con la finalidad de obtener el dato de conteo vehicular de los mismos	Obtener conteo	Concluido	6	sprint 1	esta historia de usuario contempla 2 criterios de aceptación
HUO004	Como un usuario, necesito almacenar el conteo vehicular, fecha y hora de la toma de muestra, con la finalidad de disponer de estos datos cuando se necesite presentarlos	Almacenar conteo	Concluido	2	sprint 1	esta historia de usuario contempla 2 criterios de aceptación
HUU004	Como un usuario, necesito que los usuarios se loguen especificando su rol de usuario, con la finalidad de proveer a cada usuario un menú de acceso diferente dependiendo de su rol en el sistema	logueo	Concluido	3	sprint 4	esta historia de usuario contempla 2 criterios de aceptación
HUU005	Como un usuario, necesito cargar en tablas los datos de los usuarios, cámaras, roles, relaciones, etc. En vistas diferentes con la finalidad de proveer a los usuarios dependiendo de su rol de acceso los menús correspondientes, tablas informativas de los usuarios, cámaras, roles, relaciones, etc.	cuenta usuario	Concluido	2	Sprint 4	esta historia de usuario contempla 2 criterios de aceptación

2.3 DISEÑO

En esta sección se determina el funcionamiento del sistema, además se describirá los componentes propuestos con el uso de diferentes diagramas UML.

2.3.1 ARQUITECTURA DEL PROTOTIPO

En la Figura 2.4 se observa los módulos que forman parte del prototipo que se está diseñando.

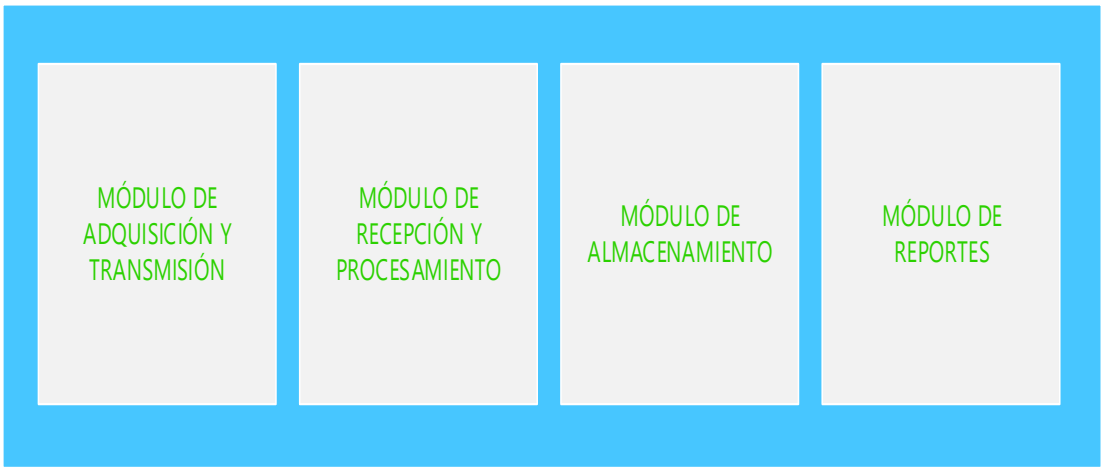


Figura 2.4 Arquitectura del prototipo

Módulo de adquisición y transmisión de video. – Este módulo está constituido por cámaras IP que se encargan de capturar y enviar el video por streaming a través de Internet hacia el módulo de recepción y procesamiento. Cabe señalar que se emplea un servicio de streaming propio de las cámaras utilizadas.

Módulo de recepción y procesamiento. – Es el encargado de recibir el video transmitido por streaming desde las cámaras e implementar un algoritmo para el conteo de vehículos en tiempo real.

Módulo de almacenamiento. – Este módulo constará de una base de datos que almacena la información de las cámaras, usuarios del sistema, el reporte del conteo de automóviles en un intervalo de tiempo determinado, fecha y hora de conteo, entre otras.

Módulo de vistas y reportes. – Contiene un conjunto de formularios que permiten realizar las operaciones CRUD (Create-Read-Update-Delete) de perfiles, usuarios y cámaras. Además, Permite mostrar la información del estado de flujo vehicular y reportes históricos.

2.3.2 FUNCIONAMIENTO DEL PROTOTIPO

En la Figura 2.5 se observa el diagrama de secuencia general del prototipo propuesto.

El funcionamiento del prototipo depende del rol de usuario que realice la autenticación en el aplicativo. Con esto la parte de autenticación es muy significativa, ya que se plantea el uso de menús dinámicos por cada rol, los mismos que estarán de acuerdo a las actividades que los roles realizaran dentro del sistema.

Para el rol de administrador se empieza con el proceso de autenticación, luego del cual se desplegarán los menús correspondientes a todas las actividades que puede desempeñar.

La actividad que presta el mayor valor para el administrador es la gestión de los datos de usuario, sobre estos el administrador podrá realizar las operaciones CRUD que considere necesarias.

Para el rol operador de cámaras después del proceso de autenticación, la actividad que tiene mayor valor para este es la gestión de cámaras y sus parámetros, sobre los cuales podrá realizar las operaciones CRUD que considere necesarias.

Con esto se puede dar paso al guardado del conteo vehicular obtenido del procesamiento del stream de una cámara siempre y cuando el operador ya haya registrado la misma en el sistema. El rol de usuario solo con el proceso de autenticación ya tendrá acceso al conteo vehicular producido por cada cámara que estarán georreferenciadas sobre un mapa. Además contara con una opción para solicitar gráficos de barras del conteo vehicular vs el tiempo dependiendo de la fecha.

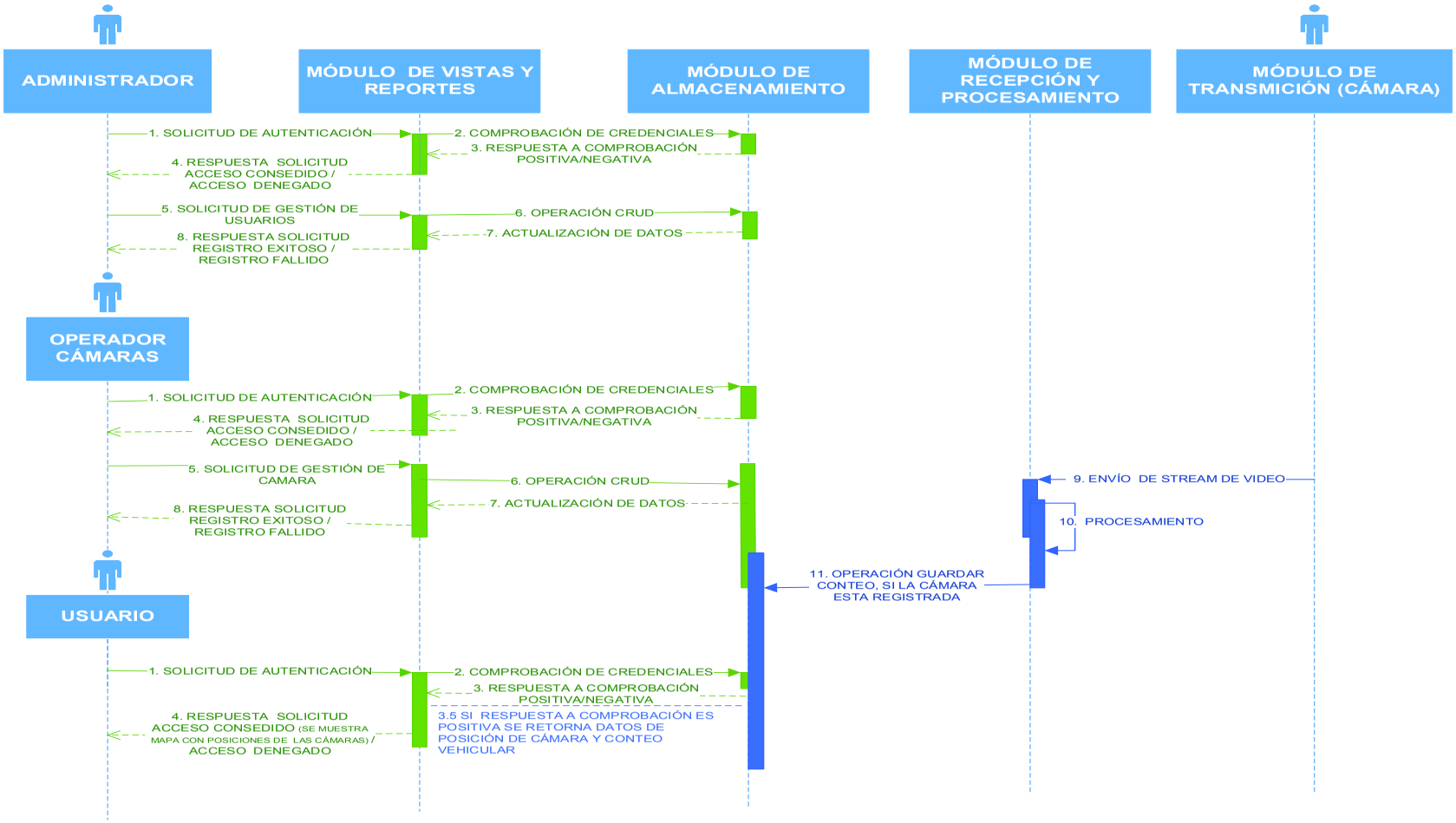


Figura 2.5 Diagrama de secuencia del prototipo

2.3.3 MÓDULO DE AQUISICIÓN Y TRANSMISIÓN DE VIDEO

Este módulo estará constituido por cámaras IP que se encargan de capturar y enviar el video por streaming a través de Internet o red Interna hacia el módulo de recepción y procesamiento.

De los resultados obtenidos en el estado del arte realizado en la sección 1.3 se debe considerar los parámetros intrínsecos mostrados en la Tabla 2.20.

Tabla 2.20 Parámetros intrínsecos de la cámara

Característica	Valor
Resolución	VGA o superior
Tipo	bala
Soporte RTSP[2]	si

Los parámetros extrínsecos mínimos deseables para las cámaras serán las que se muestran en la Tabla 2.21.

Tabla 2.21 Parámetros extrínsecos de la cámara

Característica	Valor
Altura	3 a 5 metros
Angulo	45 a 90 grados por debajo del eje x

2.3.4 MÓDULO DE RECEPCIÓN Y PROCESAMIENTO

Este módulo recibirá el video transmitido por streaming desde las cámaras e implementa un algoritmo para el conteo de vehículos en tiempo real. En la Figura

[2] RTSP. – Real Time Streaming Protocol, protocolo de capa aplicación no orientado a conexión, cuyo objetivo es controlar la entrega de datos en tiempo real.

2.6 se puede observar el diagrama de flujo general del algoritmo de conteo vehicular.

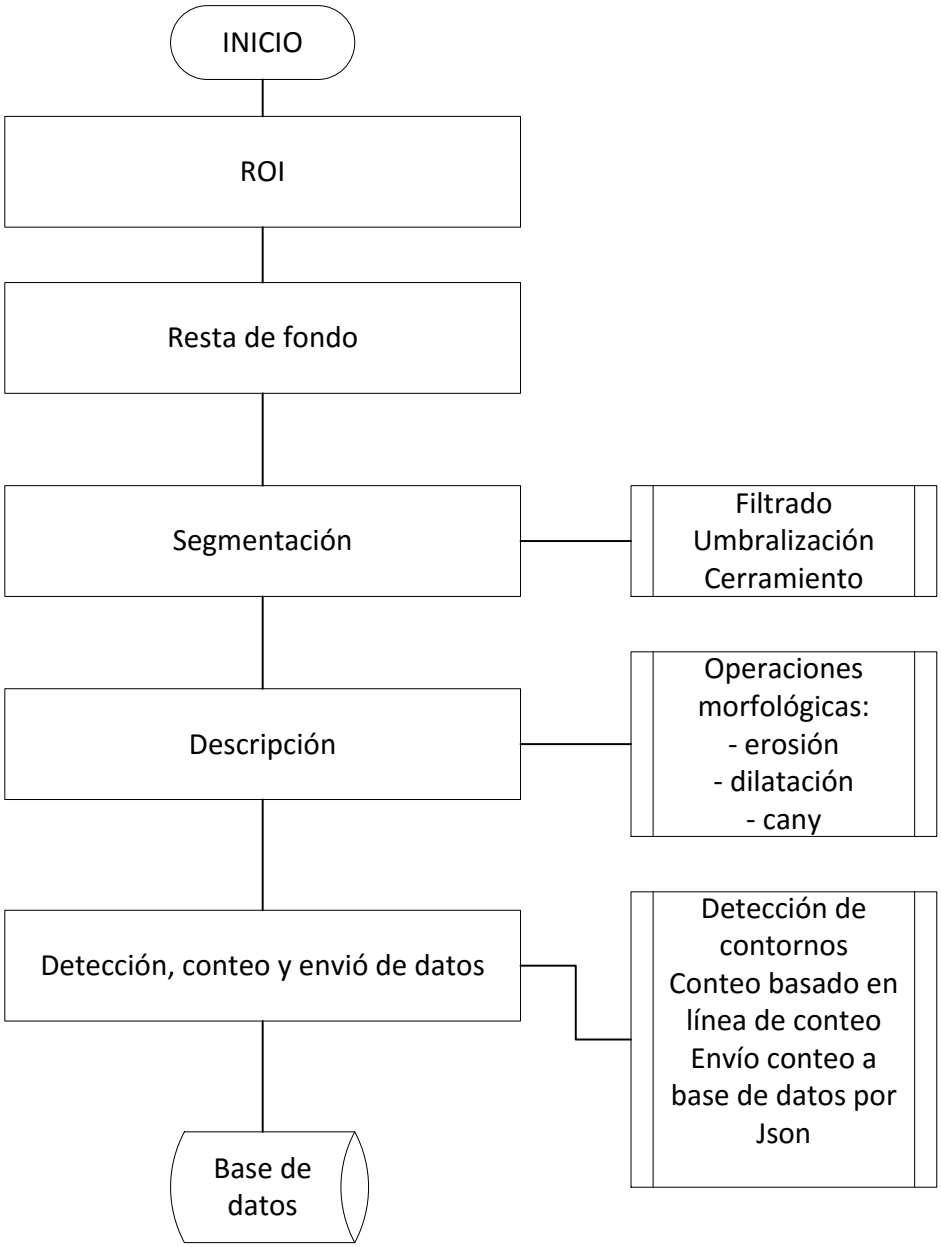


Figura 2.6 Diagrama de flujo de conteo vehicular

2.3.4.1 Algoritmo de conteo vehicular

A continuación se describe cada una de las fases que tendrá el diseño del algoritmo de conteo vehicular.

Áreas de interés (ROI)

En esta parte se establecerá el área de interés sobre el stream de video, al cual se aplicará el resto del algoritmo. Con el fin de no procesar áreas del video que no influyen en nada en el conteo vehicular y más bien generan un exceso de procesamiento. Esta idea fue tomada del análisis del estado del arte del proyecto descrito en 1.3.3. En la Figura 2.7, se muestra el diagrama de flujo de esta fase.

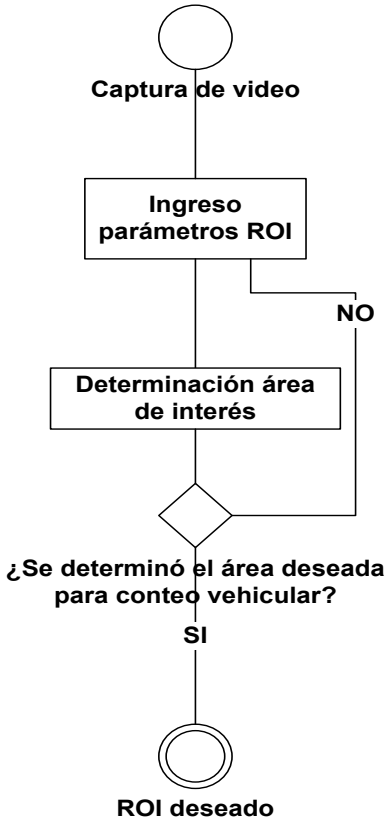


Figura 2.7 Diagrama de flujo determinación área de interés

Resta de fondo

Es un método utilizado para detectar objetos en movimiento en cámaras estáticas, consiste en restar los objetos del segundo plano o fondo del primer plano (objetos con movilidad). Esto genera una imagen en blancos y grises, los pixeles clasificados como fondo presentan un valor de gris igual a 0 y los pixeles de los vehículos en movimiento están entre 1 y 255. Este paso esta referenciado en el análisis del estado del arte descrito en la sección 1.3. En los tres proyectos analizados toman

esto como punto de partida para sus respectivos algoritmos. En la Figura 2.8 se muestra el diagrama de flujo de esta fase.

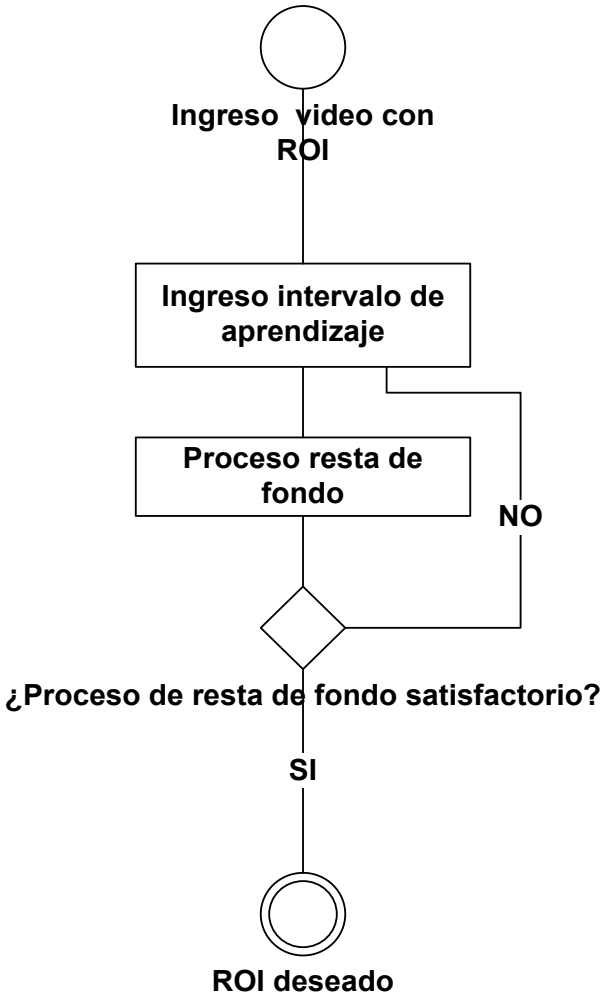


Figura 2.8 Diagrama de flujo fase resta de fondo

Segmentación

En esta parte se aplicará una serie de filtros al resultado anterior para eliminar el ruido y dejar visibles solo los objetos en movimiento. Inmediatamente se realiza la umbralización que reside en dar un valor de 255 (1 lógico) a todos los pixeles referentes al primer plano cuyo valores de grises se encuentran entre 1 y el umbral establecido y de 0 (0 lógico) a los pixeles pertenecientes al fondo. Con lo cual solo los objetos en movimiento serán de color blanco y el fondo será negro. La umbralización es un paso obligado después de la resta de fondo, que de igual forma

es ocupado encada uno de los tres proyectos analizados en 1.3. En la Figura 2.9 se muestra el diagrama de flujo de esta fase.

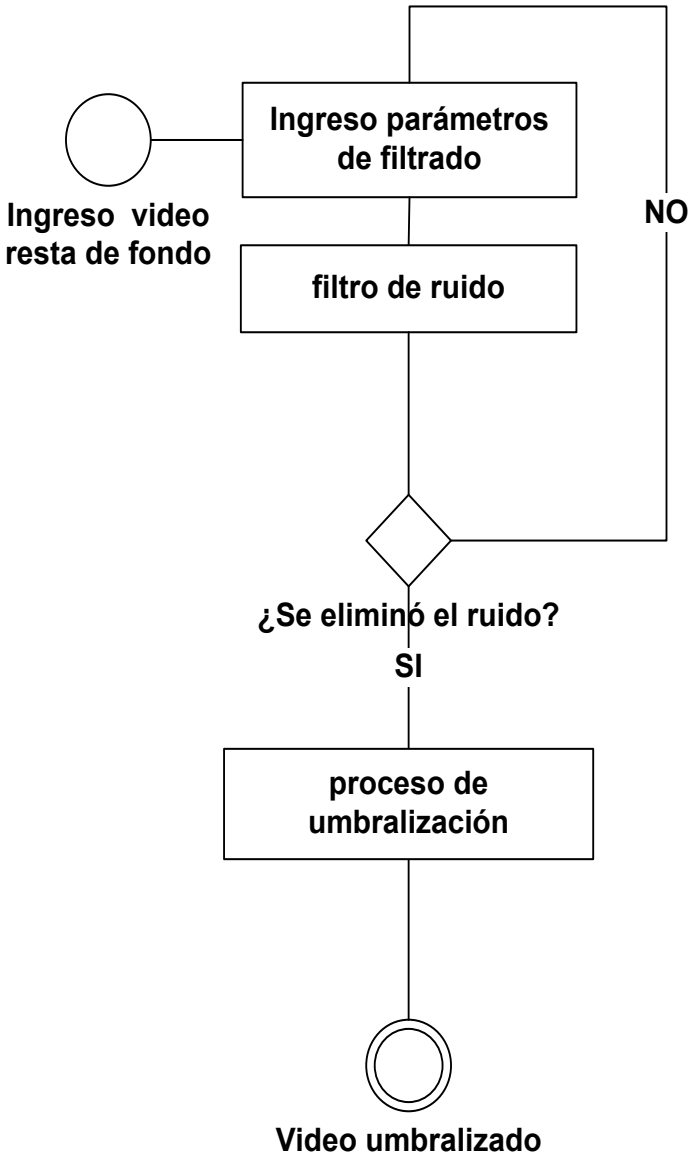


Figura 2.9 Diagrama de flujo fase de umbralización

Descripción

En esta sección se aplicará una serie de operaciones morfológicas sobre lo objetos en movimiento. Con la finalidad de formar un solo cuerpo uniforme para que pueda ser detectado. La idea fue tomada del análisis del estado del arte del proyecto descrito en 1.3.1. En la Figura 2.10 se muestra el diagrama de flujo de esta fase.

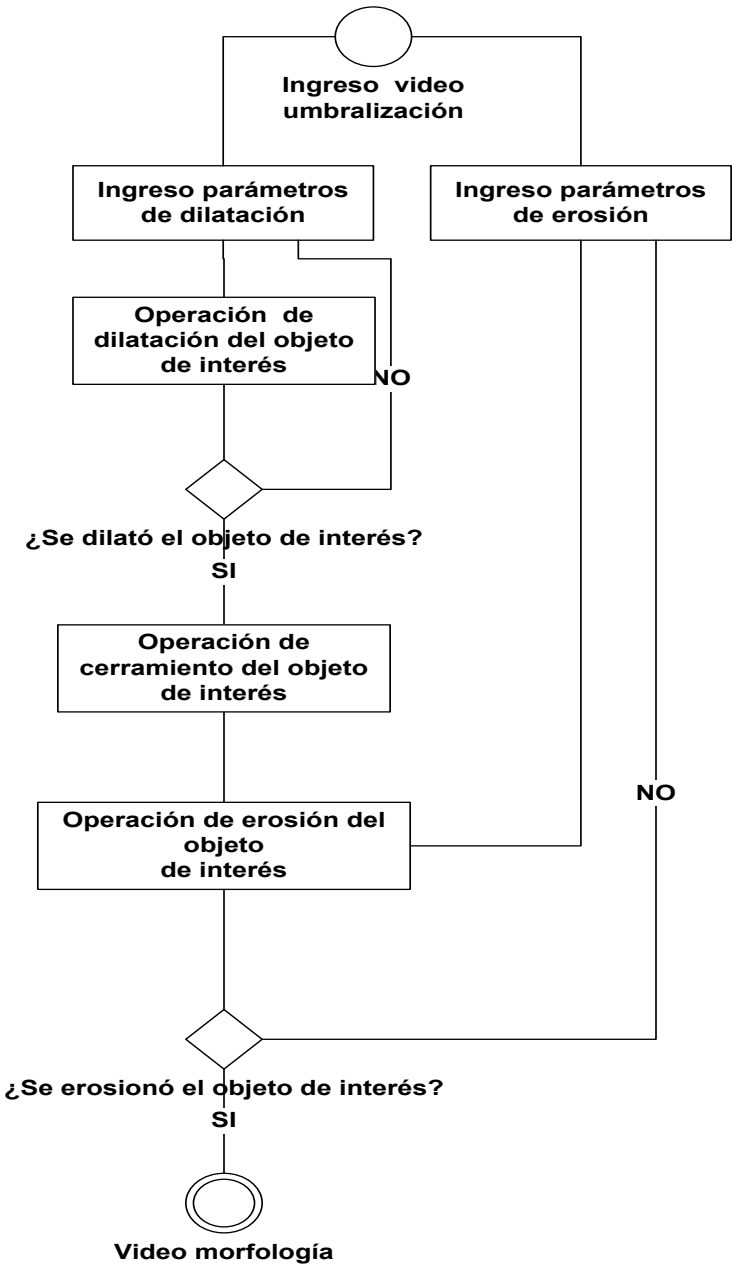


Figura 2.10 Diagrama de flujo fase de descripción

Detección, conteo, envió de datos

Para la detección de los objetos en movimiento se determinó utilizar la detección basada en contornos. Con ello se tendrá la detección del objeto y se hará uso de una línea de conteo con la que se conseguirá la información que se necesita y a la vez enviarla a guardar a la base de datos. En la Figura 2.11 se muestra el diagrama de flujo de esta fase.

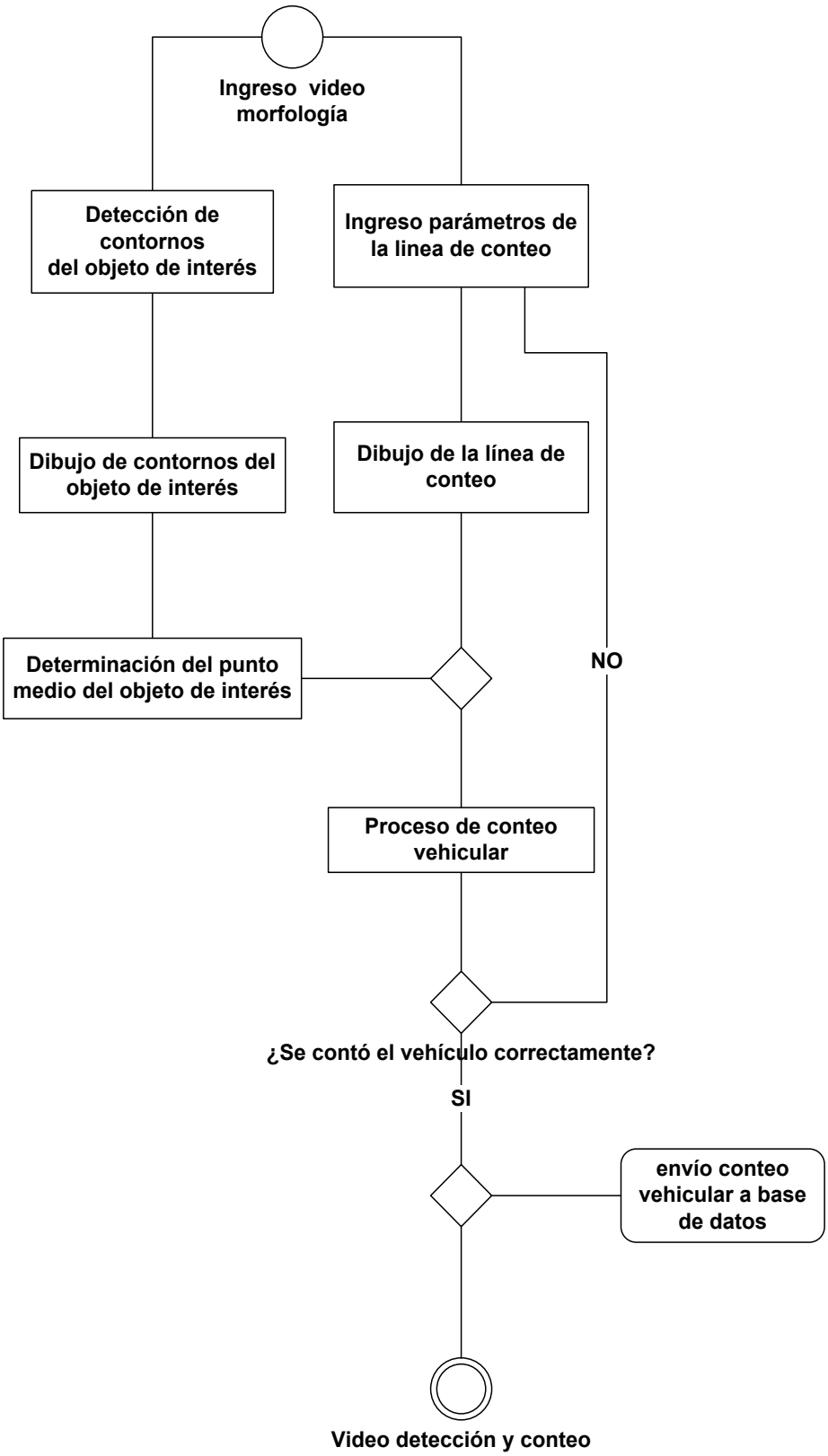


Figura 2.11 Diagrama de flujo fase detección y conteo

2.3.5 MÓDULO DE ALMACENAMIENTO

Este módulo constará de una base de datos relacional que almacenará la información de las cámaras, usuarios del sistema, el reporte del conteo de automóviles en un intervalo de tiempo determinado, fecha y hora.

2.3.5.1 Esquema de base de datos

El diseño de la base de datos para el prototipo contempla que el nombre de la base será “opencvdb” y constará de dos esquemas. Un esquema para seguridad con el fin de ofrecer un menú dinámico a los usuarios de acuerdo a los permisos del rol que desempeñan. Y un esquema de aplicación con el objetivo de desvincular los datos de la aplicación de los roles del sistema.

2.3.5.1.1 Esquema de seguridades

Este esquema maneja todas las tablas correspondientes al control de acceso. Entre los datos que maneja se encuentran perfiles, usuarios y menús. En la Figura 2.12 se puede observar el respectivo diagrama y sus tablas

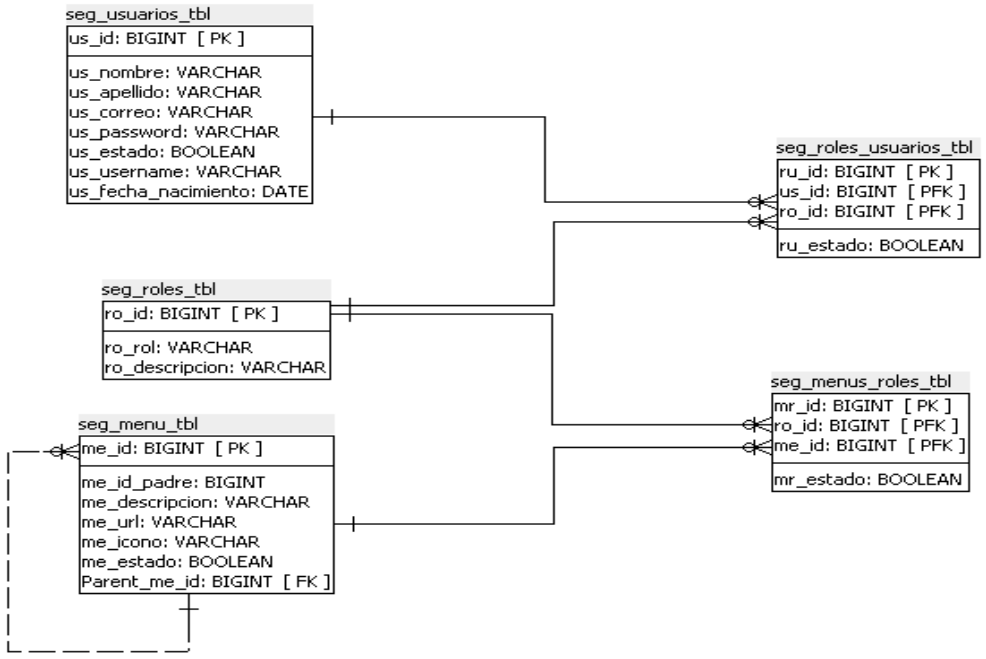


Figura 2.12 Diagrama del esquema de seguridad

Como se observa en la Figura 2.12, este esquema consta de 5 tablas, 3 son de datos y 2 son tablas de relaciones.

- **Tabla de usuarios (seg_usuarios_tbl). –** Esta tabla maneja todos los parámetros referentes al usuario de la aplicación, los cuales se describen en la Tabla 2.22.

Tabla 2.22 Parámetros de la tabla usuarios

Parámetro	Descripción
id	Identificación asignada al usuario automáticamente
Nombre	Nombre perteneciente al usuario registrado
Apellido	Apellido perteneciente al usuario registrado
Correo	Dirección de correo electrónico del usuario
Password	Credencial solicitada al usuario
Username	Credencial solicitada al usuario
Estado	Estado de usuario dentro de la aplicación
Fecha de nacimiento	Fecha de nacimiento correspondiente al usuario registrado

- **Tabla roles (seg_roles_tbl). –** Esta tabla contiene todos los parámetros referentes a los roles de usuario en la aplicación, los cuales se describen en la Tabla 2.23.

Tabla 2.23 Parámetros de la tabla roles

Parámetro	Descripción
id	identificación automática del rol de usuario
Rol	rol de usuario del sistema
Descripción	Descripción del rol y su función

- **Tabla menús (seg_menu_tbl). –** Esta tabla contiene todos los parámetros referentes a los menús a los cuales tendrá acceso el usuario en la aplicación. Se define un parámetro recursivo sobre la tabla, estos parámetros se describen en la Tabla 2.24.

Tabla 2.24 Parámetros de la tabla menú

Parámetro	Descripción
id	Identificación automática del menú
id padre	Identificación de un menú padre
Descripción	descripción de la función del menú
Url	Url de la página correspondiente al menú
Icono	Icono correspondiente al menú
Estado	Estado del menú

Tabla menús-roles (seg_menus_roles_tbl). – Esta tabla manejará las relaciones entre los roles de usuario con los menús a los que se tendrá acceso, sus campos se describen a continuación en la Tabla 2.25.

.**Tabla 2.25** Parámetro de la tabla menús-roles

Parámetro	Descripción
id	Identificación de la relación
id rol	Identificación del rol de usuario
id menú	Identificación del menú de acceso
Estado	Estado de la relación en el sistema

Tabla roles-usuario (seg_roles_ususarios_tbl). – Esta tabla manejará las relaciones entre los usuarios del sistema y los roles que les serán asignados, sus campos se describen a continuación en la Tabla 2.26.

Tabla 2.26 Parámetros de la tabla roles - usuarios

Parámetro	Descripción
Id	Identificación de la relación
Rol id	Identificación del rol de usuario
Usuario id	Identificación del usuario
Estado	Estado de la relación en el sistema

2.3.5.1.2 Esquema de datos

Este esquema maneja todas las tablas correspondientes a los datos de la aplicación de conteo vehicular. Los datos que almacena son los correspondientes a las cámaras, parámetros de configuración, flujo vehicular, entre otros. Como se observa en la Figura 2.13 este esquema consta de 5 tablas.

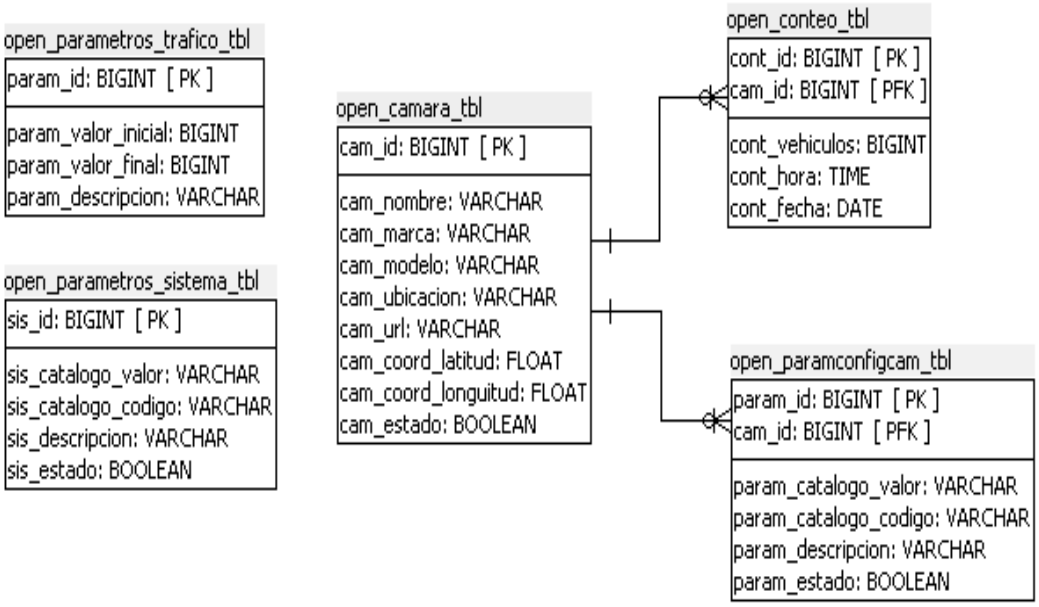


Figura 2.13 Diagrama del esquema aplicación

- **Tabla de cámaras (open_camara_tbl). –** Esta tabla contiene todos los parámetros referentes a las cámaras registradas en el sistema, estos parámetros se describen en la Tabla 2.27.

Tabla 2.27 Parámetros de la tabla cámara

Parámetro	Descripción
id	Identificación asignada a la cámara automáticamente
Nombre	Nombre perteneciente a la cámara registrada
Modelo	Modelo de la cámara registrada
Marca	Marca de la cámara registrada
Ubicación	Ubicación en la cual se instaló la cámara
Url	Url por la cual la cámara envía el stream en tiempo real
Estado	Estado de la cámara dentro de la aplicación
Coordenada latitud	Coordenada de latitud para poder geo localizar la cámara en el mapa
Coordenada longitud	Coordenada de longitud para poder geo localizar la cámara en el mapa

- **Tabla de conteo vehicular (open_conteo_tbl). –** Esta tabla contiene todos los parámetros referentes al conteo vehicular. Estos parámetros se describen en la Tabla 2.28.

Tabla 2.28 Parámetros de la tabla conteo

Parámetro	Descripción
id	Identificación asignada al conteo automáticamente
Cámara id	Id de la cámara de donde se obtiene el conteo
Conteo	Conteo vehicular
Fecha	Fecha en la que se tomó el conteo
Hora	Hora en la que se tomó el conteo

- **Tabla de configuraciones de la cámara (open_paramconfigcam_tbl). –** Esta tabla contiene todos los parámetros referentes a los parámetros de

configuración de la cámara y del procesamiento del stream. Estos parámetros se describen en la Tabla 2.29.

Tabla 2.29 Parámetros de la tabla parámetros de configuración cámara

Parámetro	Descripción
id	Identificación asignada al parámetro automáticamente
Cámara id	Id de la cámara de donde se obtiene el conteo
Catalogo valor	Valor del parámetro de configuración
Catalogo código	Código referencial del parámetro de configuración
Descripción	Descripción del parámetro de configuración
Estado	Estado del parámetro de configuración

- **Tabla de parámetros de tráfico (open_parametros_trafico_tbl). –** Esta tabla contiene todos los parámetros referentes al estimado de tráfico vehicular. Estos parámetros se describen en la Tabla 2.30.

Tabla 2.30 Parámetros de la tabla parámetros trafico

Parámetro	Descripción
id	Identificación asignada al parámetro de tráfico automáticamente
Valor inicial	Valor para determinar un límite inicial para un intervalo determinado
Valor final	Valor para determinar un límite final para un intervalo determinado
Descripción	Describe el estimado de tráfico vehicular según el intervalo asignado

- **Tabla parámetros sistema (open_parametros_sistema_tbl). –** Esta tabla contiene todos los parámetros referentes a los parámetros de configuración del sistema. Estos parámetros se describen en la Tabla 2.31.

Tabla 2.31 Parámetros de la tabla parámetros sistema

Parámetro	Descripción
id	Identificación asignada al parámetro del sistema automáticamente
Catalogo valor	Valor del parámetro de configuración del sistema
Catalogo código	Código referencial del parámetro de configuración del sistema
Descripción	Descripción del parámetro de configuración del sistema
Estado	Estado del parámetro de configuración del sistema

2.3.6 MÓDULO DE REPORTES

Para este módulo se diseñará una aplicación web con un conjunto de formularios que permitan realizar las operaciones CRUD (Create-Read-Update-Delete) de usuarios y cámaras que formarán parte del sistema. Inicialmente se manejará tres perfiles de acceso: administrador, operador y un usuario; aunque el sistema permitirá agregar más perfiles.

Además, este módulo mostrará reportes históricos del conteo vehicular que se obtiene del módulo de recepción y procesamiento. Se empleará un mapa online para presentar la información del conteo vehicular de cada cámara del sistema.

2.3.6.1 Diagrama de clases

El diagrama de clases del módulo de vistas y reportes es el que se propone en la Figura 2.14, este guarda bastante parecido con las tablas de la base de datos en general. Se plantea obtener las representaciones en objetos de cada tabla, con sus respectivas relaciones para facilitar la implementación del módulo.

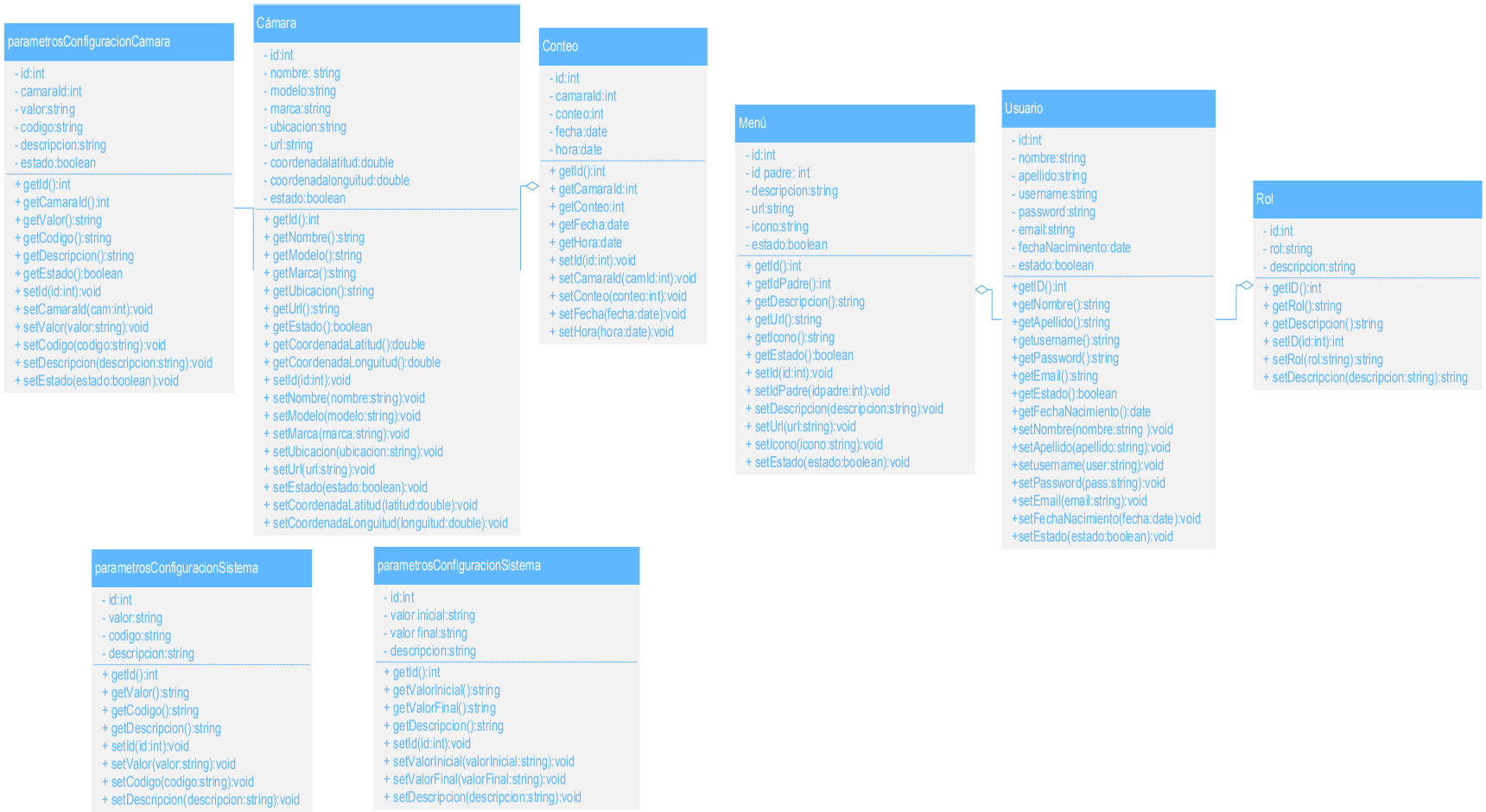


Figura 2.14 Diagrama de clases módulo de reportes

2.3.6.2 Procesos del sistema

A continuación se detalla los principales procesos en los cuales existe una interacción entre el usuario y el sistema mediante diagramas de secuencia.

2.3.6.2.1 Proceso de autenticación de usuario

El proceso de autenticación de usuario es descrito en la Figura 2.15.

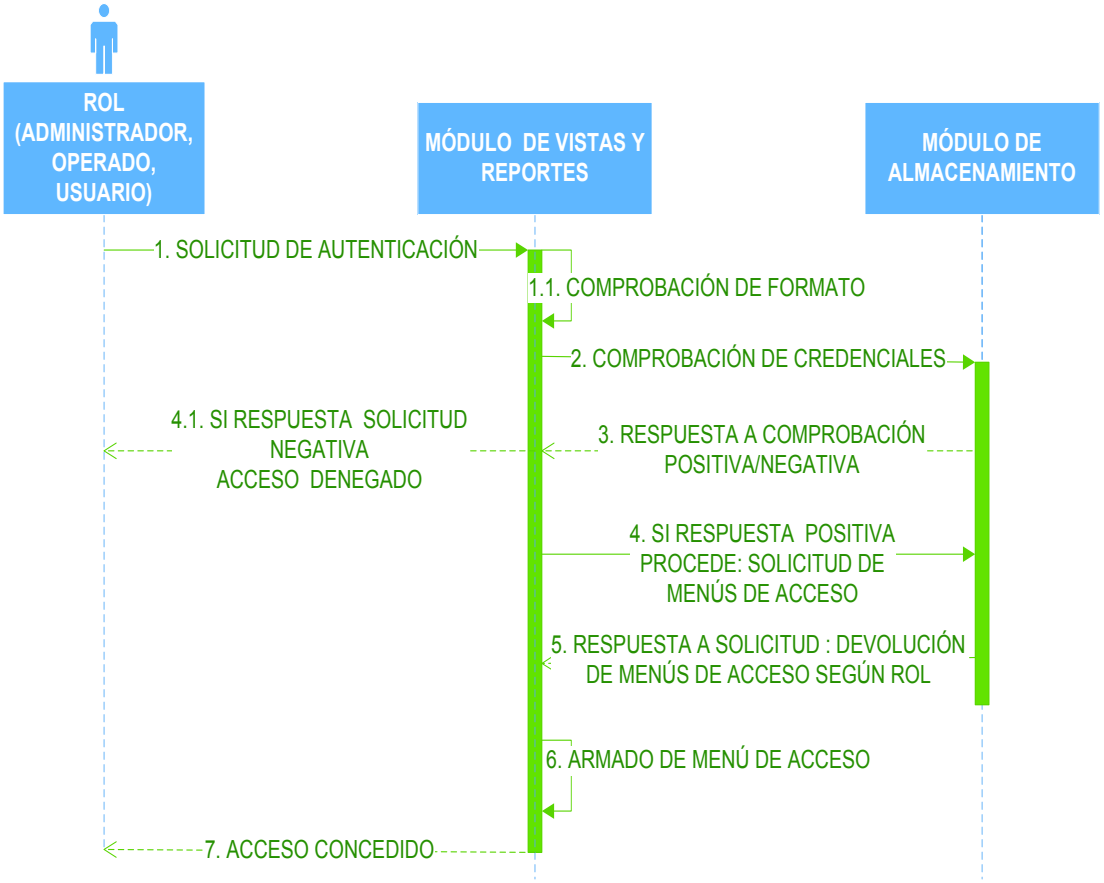


Figura 2.15 Diagrama de secuencia para autenticación

Este proceso comienza con la solicitud de ingreso por parte del usuario. El sistema pedirá las credenciales que son el username, password y rol de usuario. De inmediato se realizará la validación de formato, la cual consiste en verificar que no se tenga campos vacíos, en caso de tener dichos campos, falla en la validación y se devolverá un mensaje de error. Caso contrario se realiza la validación de las

credenciales, de no existir datos en la base o si estos son erróneos se producirá un mensaje de error. Si la validación es exitosa se recuperan las opciones de menú asociadas al usuario y se dará acceso al aplicativo.

2.3.6.2.2 Proceso para la gestión de usuarios y cámaras

En la Figura 2.16 y Figura 2.17 se encuentras los correspondientes diagramas de secuencia para la gestión de usuarios y en la Figura 2.18 y Figura 2.19 están los diagramas para la gestión de cámaras.

El proceso empieza cuando el usuario Administrador genera una solicitud de gestión de usuarios o cámaras. Se envían los datos necesarios y se valida que se hayan ingresado los campos obligatorios, si es exitosa pasará al siguiente paso, caso contrario el sistema pedirá corrección y mostrará un mensaje de error. A continuación si no hubo problemas con la validación se realiza la consulta a la base de datos, ya sea para crear, actualizar o eliminar los datos. En el caso de ser exitosa la consulta se da una respuesta de registro exitoso al usuario.

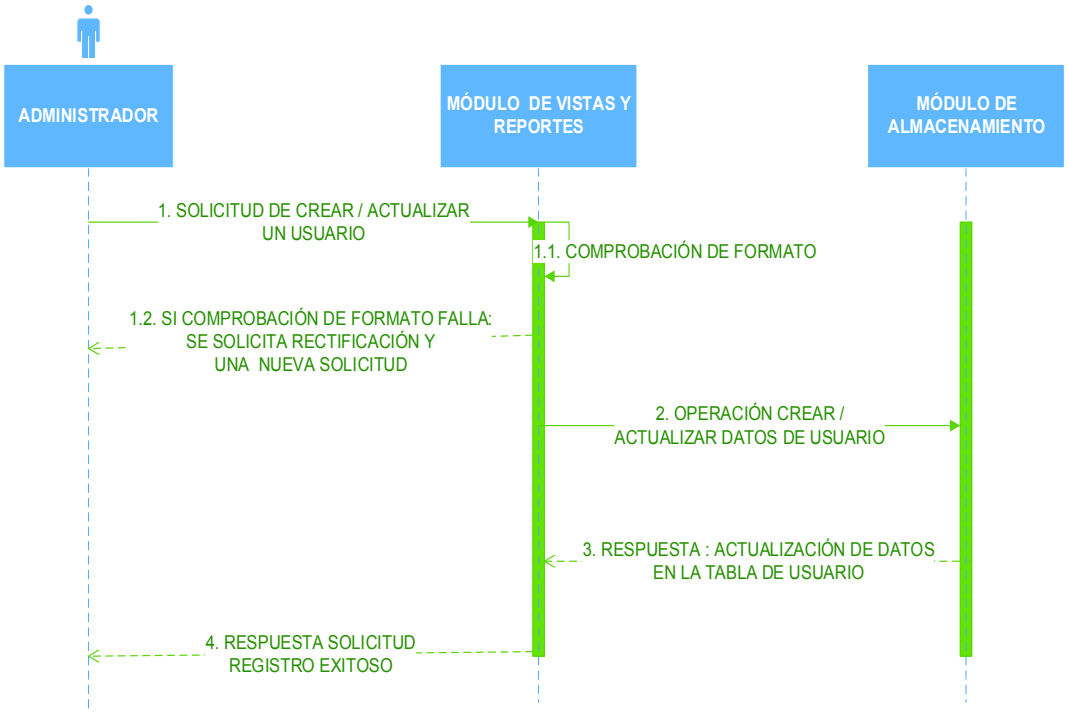


Figura 2.16 Diagrama de secuencia para crear / actualizar usuarios

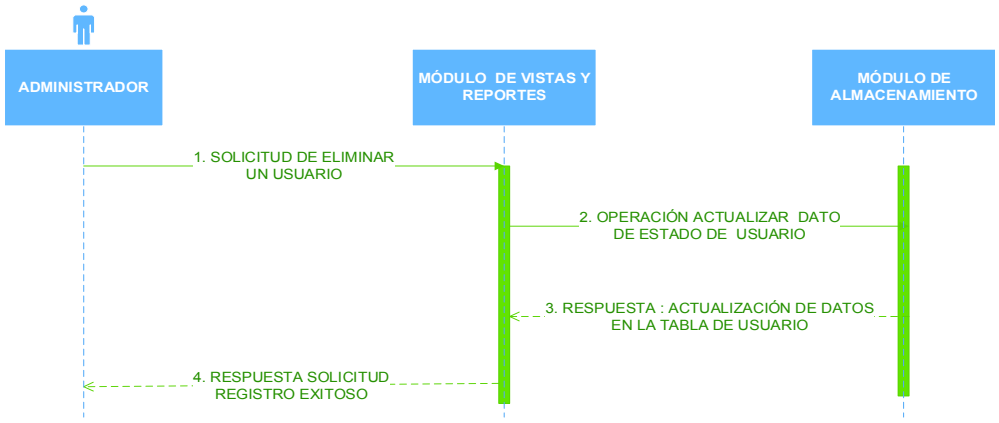


Figura 2.17 Diagrama de secuencia para eliminar usuarios

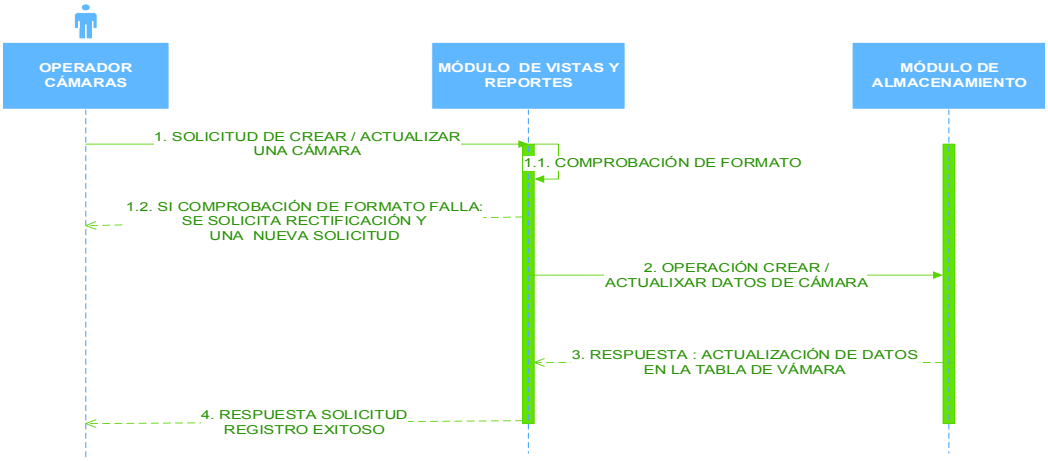


Figura 2.18 Diagrama de secuencia para crear /actualizar cámaras

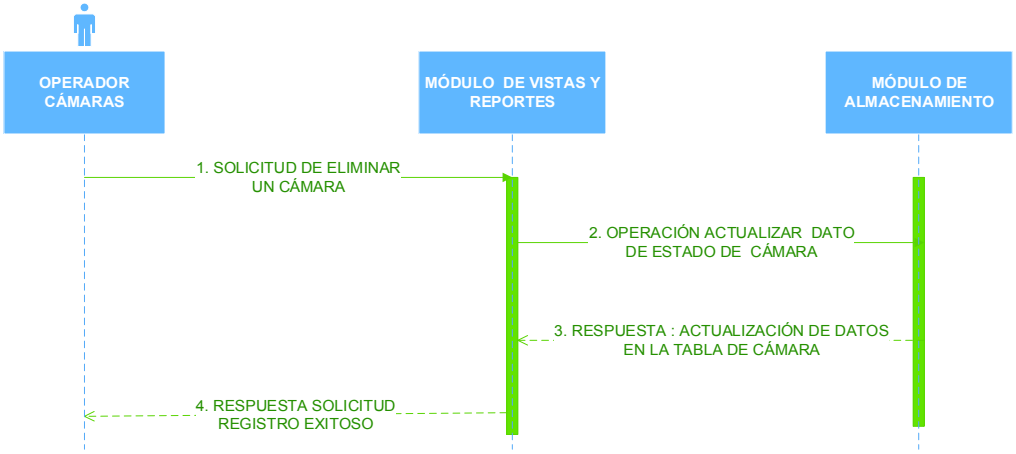


Figura 2.19 Diagrama de secuencia para eliminar de cámaras

2.3.6.2.3 Proceso de generación de reportes

El proceso de generación de reportes sobre el mapa se muestra en Figura 2.20. Este procedimiento se genera al momento que el usuario se autentica e ingresar a la aplicación, el usuario dispondrá de una vista de home la cual contendrá una mapa en el cual se geo localizarán las cámaras y se podrá obtener la información de conteo vehicular, nombre, fecha y hora de toma del conteo de cada una de las mismas.

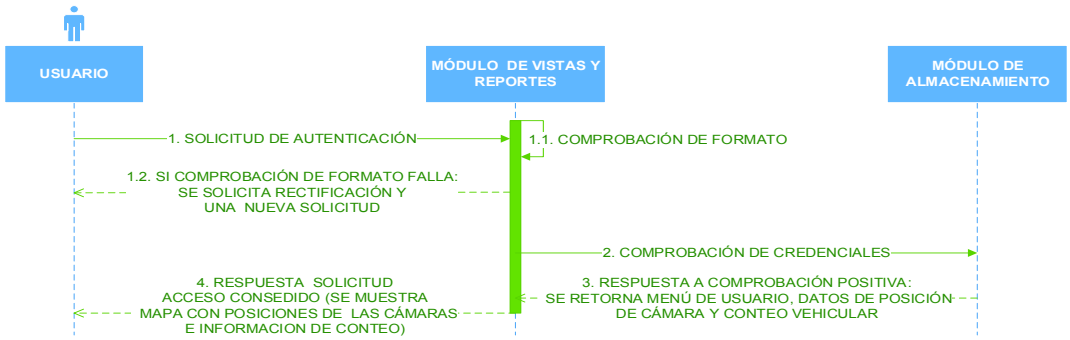


Figura 2.20 Diagrama de secuencia para generar reportes sobre mapa

El proceso de generación de reporte en gráfico de barras se muestra en la Figura 2.21, este empieza generando la solicitud. Esta petición de reporte requiere parámetros que el usuario ingresará previamente y de acuerdo con estos se generará la consulta a la base de datos. Si existe algún problema con la consulta el sistema no hará nada, si la consulta a la base fue exitosa, los datos de conteo vs tiempo serán presentados al usuario en un gráfico de barras.

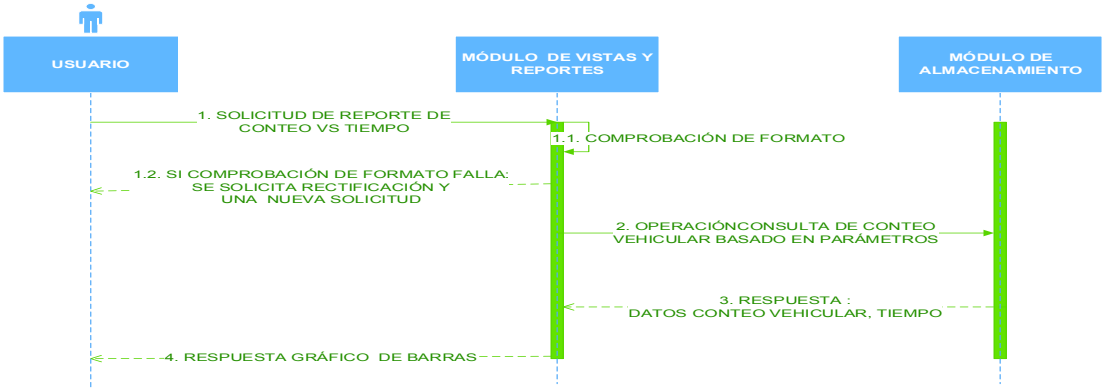


Figura 2.21 Diagrama de secuencia para generar reportes gráfico de barras

2.3.6.3 Interfaces de usuario

A continuación se presenta el prototipado de las interfaces de usuario del sistema y su navegabilidad. A continuación en la Figura 2.22, se muestra el diagrama de navegabilidad propuesto.

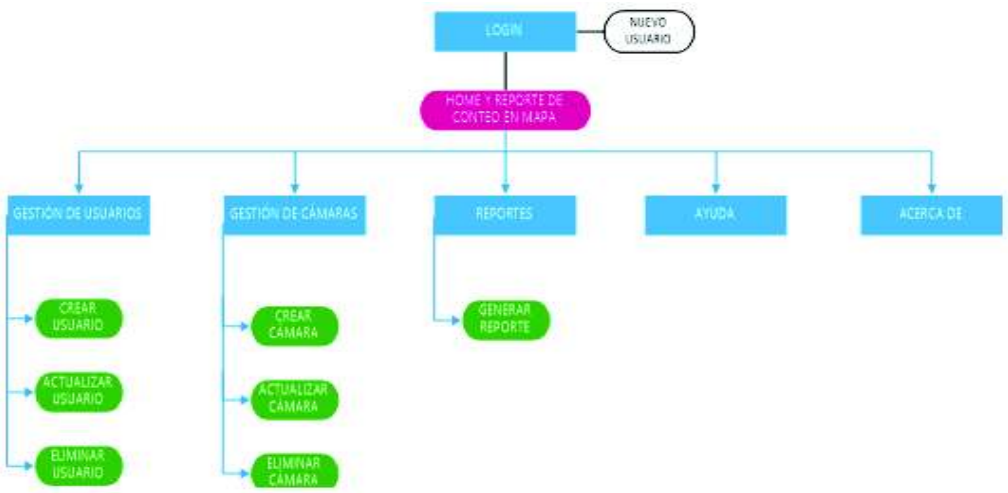


Figura 2.22 Organigrama de navegación de la aplicación

Vista de autenticación

En la vista de autenticación se solicitará el username, password y su respectivo rol de usuario, en la Figura 2.23 se presenta el prototipado de dicha vista.

Figura 2.23 Vista de autenticación

Vista de registro de nuevo usuario

A través de esta vista se crea una cuenta con un rol de usuario. No es posible crear una cuenta con un rol diferente, esta tarea la llevará a cabo el administrador. En la Figura 2.24 se encuentra el prototipado de esta vista.

Figura 2.24 Vista de registro para los usuarios nuevos

Interfaces de home y reporte de conteo

La vista de home cambiará ligeramente para cada uno de los roles de usuario en el sistema, debido a que estos tendrán privilegios diferentes. Por tal razón se diseñará un menú dinámico dependiendo del rol de usuario, un bosquejo de las tres interfaces se muestra en la Figura 2.25, Figura 2.26 y Figura 2.27.

Figura 2.25 Vista de Home para los usuarios administradores

Figura 2.26 Vista de Home para los usuarios Operadores cámaras

Figura 2.27 Vista de Home para los usuarios normales

Interfaces de gestión de usuario

Permite realizar las operaciones CRUD de los usuarios que usarán el sistema. Un bosquejo de la interfaces con la navegabilidad de cada operación CRUD se muestra en la Figura 2.28, Figura 2.29 y Figura 2.30.

Figura 2.28 Vista de crear usuario

Figura 2.29 Vista de actualización de usuarios

Figura 2.30 Vista de eliminación de usuarios

Interfaces de gestión de cámaras

La vista de gestión de cámaras en el sistema será única y se podrá realizar en esta las operaciones CRUD sobre las mismas, el prototipado de la interfaces con la navegabilidad de cada operación CRUD se muestra en la Figura 2.31, Figura 2.32

y Figura 2.33. Los datos que se podrán manipular en estas vistas se listan a continuación:

- Nombre de la cámara.
- Modelo, marca.
- Ubicación.
- Coordenadas geográficas.
- Url de conexión para capturar el stream.

Figura 2.31 Vista creación de cámaras

Figura 2.32 Vista de actualización de cámaras

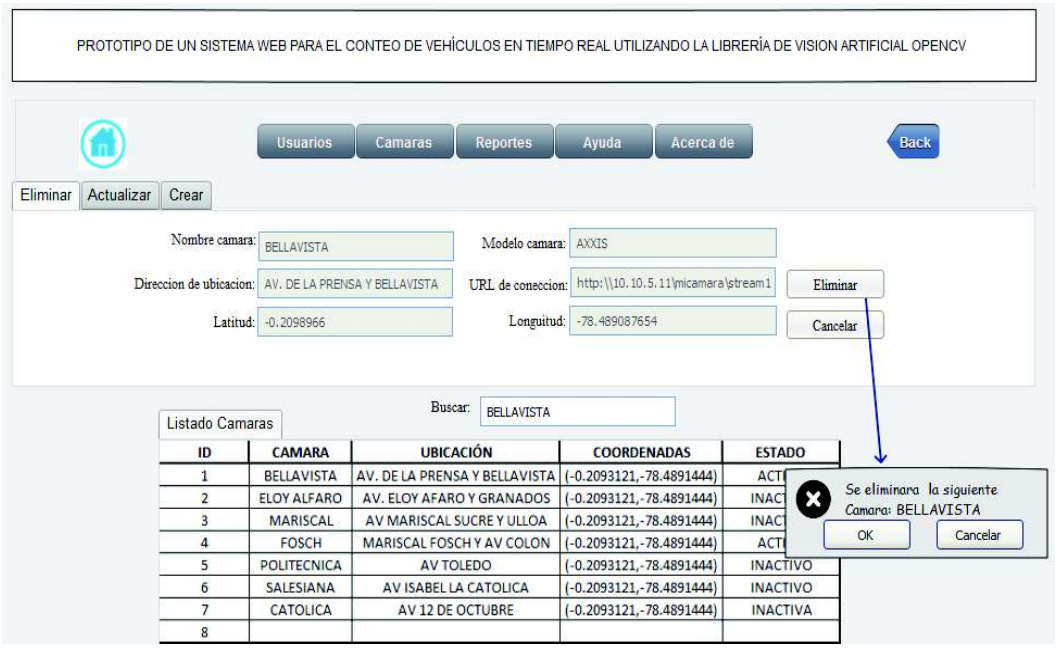


Figura 2.33 Vista de eliminación de cámaras

Vista de reportes

La vista de generación de reportes devolverá un gráfico estadístico con el flujo vehicular determinado por cada cámara. Los reportes se pueden filtrar por cámara, por hora y por día. En la Figura 2.34 se puede observar el bosquejo de esta vista.

Figura 2.34 Vista de Reportes

Vista de Ayuda

La vista de ayuda proveerá será un enlace donde se podrá descargar el manual de usuario. En la Figura 2.35 se puede observar el bosquejo de esta vista.

Figura 2.35 Vista de ayuda

Vista de Acera de

Esta vista muestra información relativa la tecnología que se utilizó en la implementación así como información del grupo encargado del proyecto. En la Figura 2.36 se puede observar el bosquejo de esta vista.

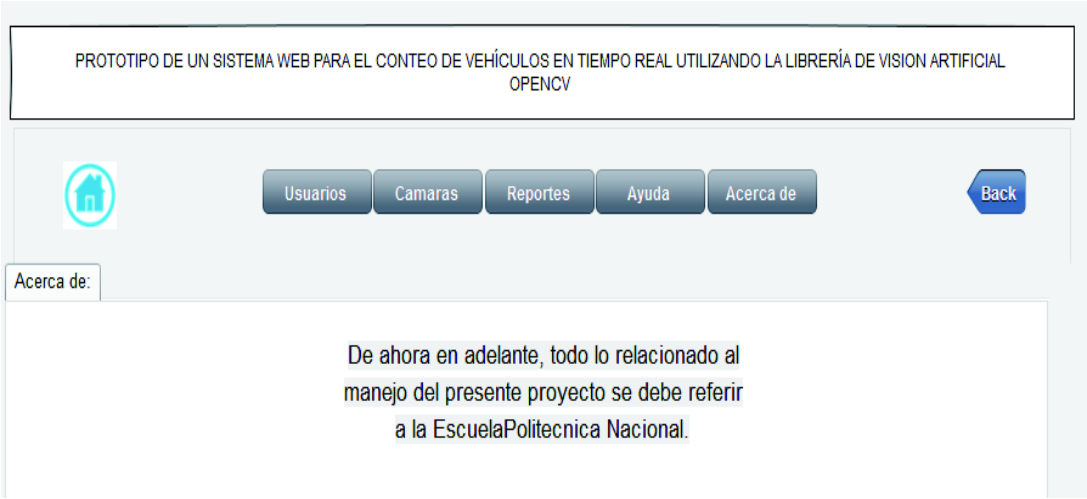


Figura 2.36 Vista de acerca de

2.4 IMPLEMENTACIÓN

En esta sección se presenta la implementación del prototipo propuesto.

2.4.1 DESPLIEGUE DEL PROTOTIPO

Un diagrama de despliegue constituye una vista estática de la configuración en tiempo de ejecución de los nodos que actúan en el proceso y de los componentes que se ejecutan en esos nodos, en el cual se puede especificar tanto el hardware como el software utilizado [22].

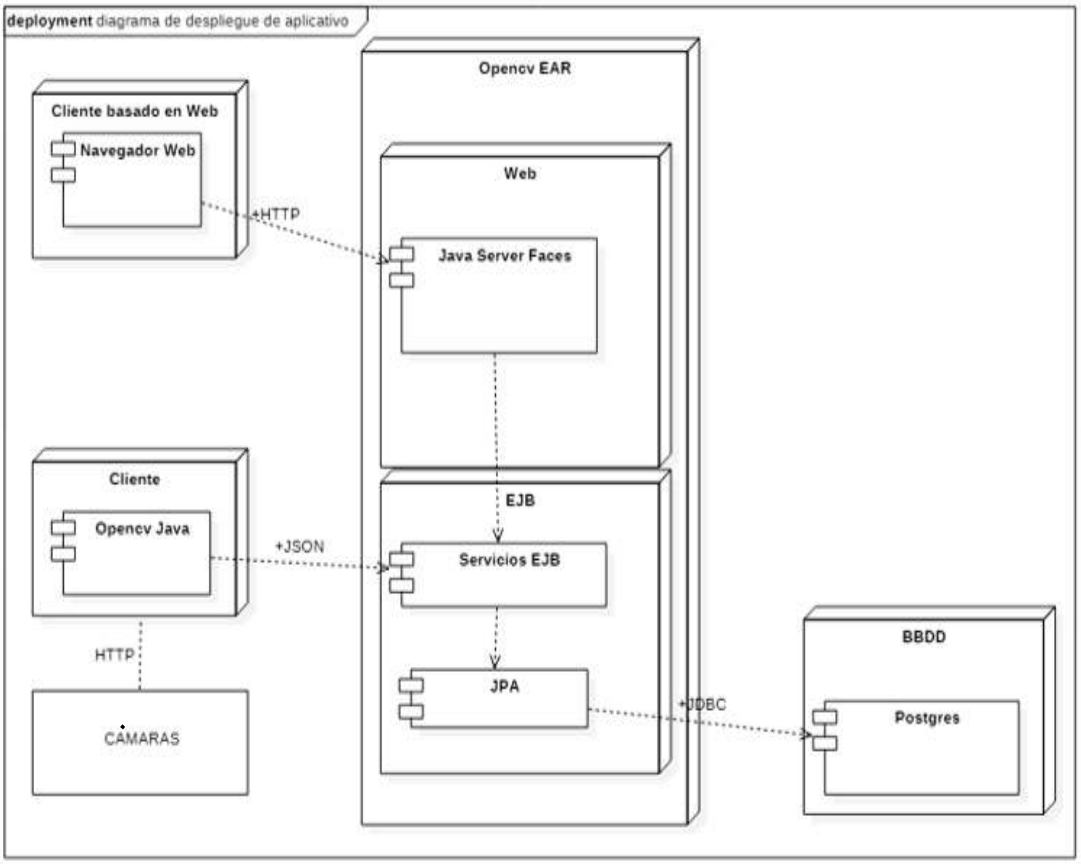


Figura 2.37 Diagrama de despliegue del prototipo

2.4.2 MÓDULO DE ADQUISICIÓN Y TRANSMISIÓN DE VIDEO

La implementación de este módulo consiste en la selección de las cámaras que cumplan con los requisitos expuestos en la sección 2.2.3.

2.4.2.1 Cámara

Las cámaras seleccionadas para la implementación de este módulo son las siguientes:

Cámara IP Trendnet TV-IP522P

Las características de esta cámara se muestran en la Tabla 2.32

Tabla 2.32 Características cámara entrada administración [23]

Image & Video Trendnet TV-IP522P	
Compresión:	simultánea H.264 / MPEG-4 / MJPEG
Perfiles:	hasta 4 perfiles simultáneos
Tipo:	Bala
Control de exposición/balance de blancos:	Automático
Resolución:	4VGA (1280x960) hasta 15fps, HDTV (1280x720) hasta 15fps, VGA (640x480) hasta 30fps
Soporte RTSP	Si

Cámara IP Dahua DH-IPC-HFW2300RN-Z

Las características de esta cámara se muestran en la Tabla 2.33

Tabla 2.33 Características cámara entrada CEC [24]

Image & Video Dahua DH-IPC-HFW2300RN-Z	
Compresión:	Simultánea H.264D / H.264 / MPEG-4
Perfiles:	2 perfiles simultáneos
Tipo:	Bala
Control de exposición/balance de blancos:	automático
Resolución:	Max:3 Mpx (2121x1414) Min; VGA (640x480) FPS: 1 a 30
Soporte RTSP Y HTTP	si

Estas cámaras se escogieron debido a su disponibilidad inmediata para este proyecto y porque cumplen los requisitos expuestos en el diseño.

2.4.3 MÓDULO DE RECEPCIÓN Y PROCESAMIENTO

2.4.3.1 Tecnologías seleccionadas

Para el desarrollo de este módulo se optó por las siguientes tecnologías.

Java. – Para el desarrollo de este módulo se utilizó el lenguaje de programación Java en vista de que posee una integración con la librería de visión artificial Opencv y a la experiencia previa con este por parte del desarrollador.

Opencv. – Para el desarrollo de este módulo se empleó las librerías y métodos contenidos en Opencv 3.0. Estas librerías están enfocadas al desarrollo de aplicaciones de visión artificial. OPENCV se compiló desde cero en la plataforma seleccionada, la guía para eso la puede encontrar en [25].

Entorno de desarrollo integrado. – El IDE escogido para el desarrollo de este módulo fue Eclipse en su última versión "Mars 1.0 para desarrollo JEE" debido a la experiencia previa del programador.

Apache maven. – Es una herramienta para gestión de proyectos y manejo de dependencias, basado en un archivo llamado POM.xml donde se define lo que necesita el proyecto. Para la implementación de este módulo se trabaja con maven 3.3.9 que es la última versión disponible y se puede encontrar en [26].

Apache archiva. – Es una herramienta para gestionar localmente un repositorio de artefactos el cual es ideal para trabajar con maven y así lograr que la descarga de dependencias sea local y no desde maven central.

Para la implementación de este módulo se está trabajando con archiva 2.2.0 que es la última versión disponible y se puede encontrar en [27].

2.4.3.2 Implementación del algoritmo

Para la implementación del algoritmo se siguió la guía que ofrece el libro de consulta "Opencv 3.0 Computer Vision with Java" [25]. El cual ofrece una introducción a los diferentes métodos que ofrece Opencv con Java.

A continuación se presentará una descripción de las partes más relevantes en la implementación.

A continuación se muestra del diagrama de flujo del algoritmo en la Figura 2.38, y de acuerdo a este está el orden de la implementación.

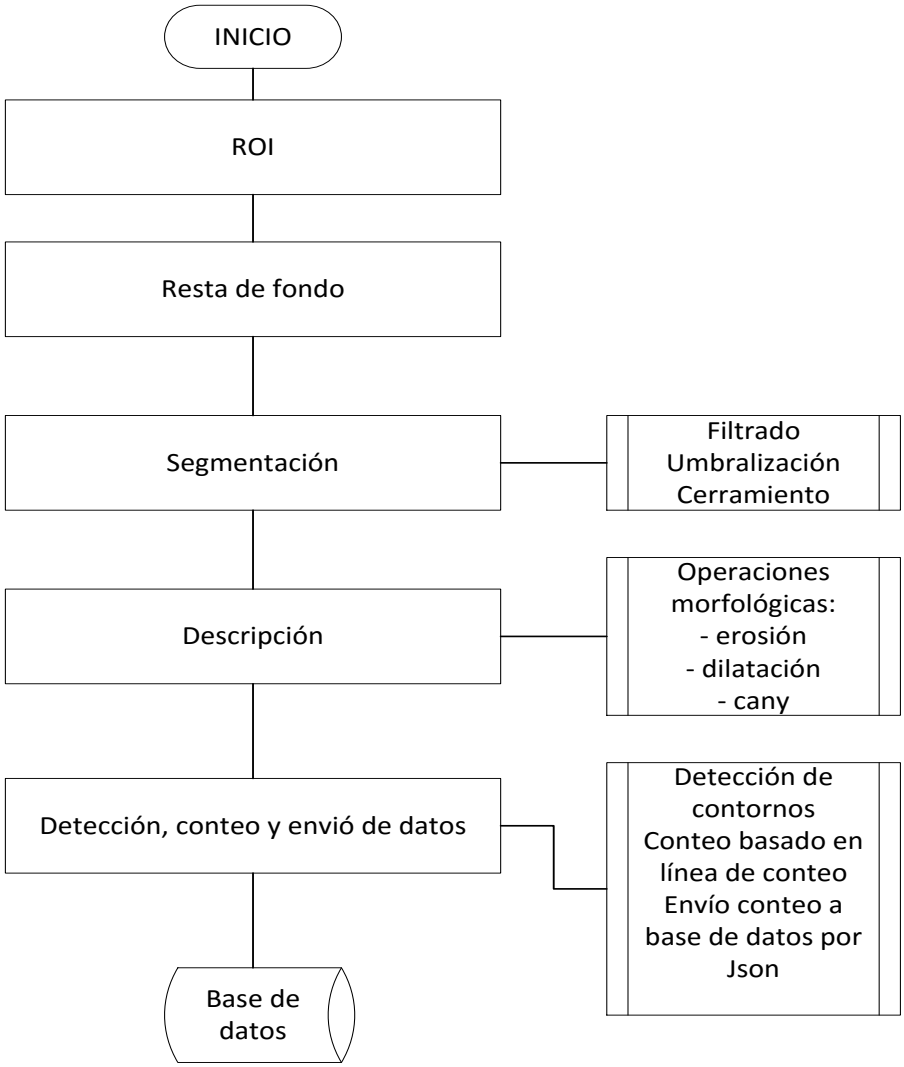


Figura 2.38 Diagrama de flujo del algoritmo

La captura de video se llevó a cabo con la clase VideoCapture de Opencv, esta tiene varias opciones de parámetros de recepción. Entre estas el path de un video pregrabado o el path del streaming de video en tiempo real producido desde una cámara IP, esta última es la característica que se utilizó en este proyecto. En el Código 2.1 se observa la línea de código correspondiente a la captura de video

```
//video a usar
VideoCapture capturaVideo = new VideoCapture(Constante.URL_VIDEO_LOCAL);
```

Código 2.1 Captura de video

Para el desarrollo del algoritmo se utilizó videos de prueba los cuales están adjuntos en el ANEXO D.

2.4.3.2.1 Visualización de video y área de interés (ROI)

Para poder visualizar la captura de video se utilizó Swing, debido a que la interfaz nativa de Opencv llamada HighGUI tenía un método para visualizar imágenes y video llamado imshow el cual estuvo disponible para Java hasta la versión 2.4.7.0 de Opencv.

La clase ProcesadorImagen contiene el método llamado toBufferedImage, el cual convierte un objeto OpenCV del tipo Mat a un tipo BufferedImage de AWT. El bloque de código del método se puede observar en el Código 2.2.

```
public BufferedImage toBufferedImage(Mat matriz){
    int tipo = BufferedImage.TYPE_BYTE_GRAY;
    if ( matriz.channels() > 1 ) {
        tipo = BufferedImage.TYPE_3BYTE_BGR;
    }
    int tamañoBuffer = matriz.channels()*matriz.cols()*matriz.rows();
    byte [] buffer = new byte[tamañoBuffer];
    matriz.get(0,0,buffer); // get all the pixels
    BufferedImage imagen = new BufferedImage(matriz.cols(),matriz.rows(), tipo);
    final byte[] pixelsObjetivo = ((DataBufferByte) imagen.getRaster().getDataBuffer()).getData();
    System.arraycopy(buffer, 0, pixelsObjetivo, 0, buffer.length);
    return imagen;
}
```

Código 2.2 Método toBufferedImage

Este método es utilizado para verificar los resultados de cada fase del algoritmo. En la figura Figura 2.39 se puede visualizar la captura de video.

Figura 2.39 Visualización de captura de video

Para obtener el área de interés sobre el video se hace uso del método subMat perteneciente a la clase Mat de Opencv, la cual recibe como parámetros lo siguiente:

- imagen.subMat(Fila inicial, Fila final, Columna inicial, columna final)
  - **Fila inicial y final. –** Definen las filas inicial y final de pixeles de la imagen, con esto se controla el alto del ROI a definir sobre la imagen original.
  - **Columna inicial y final. –** Definen las columnas inicial y final de pixeles de la imagen, con esto se controla el ancho del ROI a definir sobre la imagen original.

El código con la implementación completa se encuentra en el ANEXO C.1.

2.4.3.2.2 Resta de fondo

El proceso de resta de fondo se realiza utilizando el método BackgroungSustractorMOG2 perteneciente a la clase Video. Este recibe los siguientes parámetros:

- Video.createBackgroundSubtractorMOG2 (int history, double varThreshold, boolean detectShadows)
  - **History. –** Es la longitud de la historia (número de imágenes con las cuales se realiza el cálculo de la resta de fondo).
  - **Threshold. –** Es el umbral que se establece para considerar al fondo de color negro (0 lógico) y grises a los objetos en movimiento.
  - **detectShadows. –** Es una variable booleana que, si su valor es True, el algoritmo detecta y marca las sombras.

Para el presente proyecto se usa el constructor sin parámetros, los valores por defecto son los siguientes:

- **History =** 500
- **Threshold =** Binary
- **detectShadows =** True

BackgroundSubtractorMOG2 propone una taza de aprendizaje, que determina el tiempo durante el cual se va a volver a ejecutar el algoritmo. En este intervalo se determina los objetos del primer plano (objetos en movimiento), y resta el segundo plano, en el Código 2.3 se puede observar el bloque de código correspondiente.

```
public class RestaFondo implements ProcesadorVideo {
    private BackgroundSubtractorMOG2 mog = org.opencv.video.Video.createBackgroundSubtractorMOG2();
    private Mat primerPlano = new Mat();
    private Mat temp = new Mat();
    private double tasaAprendizaje = 0.06;

    public Mat process(Mat videoEntrada) {
        mog.apply(videoEntrada, primerPlano, tasaAprendizaje);
        return primerPlano;
    }
```

Código 2.3 Clase resta de fondo

En la Figura 2.40 se puede observar el resultado de aplicar este método.

Figura 2.40 Visualización resta de fondo

2.4.3.2.3 Umbralización

Los métodos de Opencv que se utiliza son de desvanecimiento o difuminado: Blur y GaussianBlur pertenecientes a la clase Imgproc.

- **Blur. –**Se tiene una matriz de 3x3 que tiene un total de 9 pixeles y se obtiene el promedio. Con esto se logra que cada pixel de salida tenga el valor promedio de los 9 pixeles vecinos de la imagen original (difuminación). Los parámetros que recibe método son:
    - blur(Mat src, Mat dst, Size Ksize)
        - **Src. –** Imagen de entrada.
        - **Dst. –** Imagen de salida.

  - **Ksize. –** Es el tamaño de la matriz que se usa como kernel[3].
  -

- **GaussianBlur. -** La idea para el difuminado es similar al método anterior, con la única diferencia que no se utiliza el mismo valor promedio para cada pixel, sino los valores obtenidos de una curva de Gauss para así priorizar los pixeles del centro.
  - GaussianBlur(Mat src, Mat dst, Size ksize, double sigmaX [, double sigmaY])
    - **Src. –** Imagen de entrada.
    - **Dst. –** Imagen de salida.
    - **Ksize. –** Es el tamaño de la matriz que se usa como kernel.
    - **Sigma. –** Es una desviación estándar que es aproximadamente la mitad del ancho cuando él alto se encuentra a la mitad de la altura máxima de Gauss.

Esta clase recibe como parámetro la matriz Mat de resta de fondo. Como se observa en la Figura 2.40, existen puntos blancos que no pertenecen a los objetos en movimiento, por tanto estos pueden ser tratados como ruido. Por esta razón se utiliza los filtros de difuminado *GaussianBlur* y *Blur*. De inmediato se segmenta la matriz Mat obtenida, aplicando el método *Threshold,* el resultado será una imagen binaria.

Una vez aplicado el método *Threshold* se podría dar el caso que existan áreas dentro del objeto de interés, cuyos pixeles estén debajo del umbral y no sean considerados parte del objeto; por esta razón inmediatamente se le aplica el método *floodFill* que realiza un relleno de inundación. La idea es comprobar que los objetos de interés mantienen una conexión entre sus pixeles formando un cuerpo sólido. En el código 2.4 se puede observar la implementación.

---

[3] Kernel (núcleo). – Es una matriz de tamaño fijo con coeficientes numéricos y un punto de anclaje que regularmente es su centro, es utilizado para realizar operaciones como la convolución con otras matrices (imágenes).

```
Imgproc.GaussianBlur(imagenEntrada, dest, new Size(11, 11), 0);
Imgproc.blur(dest, dest1, new Size(4, 3));
Imgproc.threshold(dest1, dest2, -1, 255, Imgproc.THRESH_OTSU + Imgproc.THRESH_BINARY);
Imgproc.floodFill(dest2, new Mat(), new Point(1, 1), new Scalar(0));
return dest2;
```

Código 2.4 Métodos para umbralización

En la Figura 2.41 se observa el resultado obtenido en esta fase.

Figura 2.41 Visualización umbralización

2.4.3.2.4 Operaciones morfológicas

Los métodos de Opencv pertenecientes a la clase Imgproc utilizados en esta fase se describen a continuación:

- **Dilate. –** Es una operación de convolución entre una imagen y un kernel (matriz), a medida que el kernel escanea la imagen calcula el valor de pixel máximo común entre la imagen y el kernel. Esto hace que las regiones brillantes crezcan ya sea en alto o en ancho. A continuación se describe los parámetros que recibe esta clase.

    - Imgproc.dilate(Mat src, Mat dst, Element kernel)
        - **Src. –** Imagen de entrada.
        - **Dst. –** Imagen de salida.
        - **Ksize. –** Es la estructura elemental usada para la dilatación, esta puede ser rectangular, en forma de elipse o cruz.

    Erode. – Es una operación de convolución entre una imagen y un kernel (matriz), a medida que el kernel escanea la imagen calcula el valor de pixel mínimo común entre la imagen y el kernel. Esto hace que las regiones brillantes decrezcan ya sea en ancho o en alto. A continuación se describe los parámetros que recibe esta clase.

- Imgproc.erode(Mat src, Mat dst, Element kernel)
  - **Src. –** Imagen de entrada.
  - **Dst. –** Imagen de salida.
  - **Ksize. –** Es la estructura elemental usada para la erosión, esta puede ser rectangular, en forma de elipse o cruz.

En la clase Morfología se aplica una serie de operadores morfológicos que transforman la forma del objeto de interés. El primer método en aplicarse es dilate. De inmediato se aplica el método morphologyEx con el cual se realiza el cerramiento del objeto, logrando la independencia del mismo con respecto a otros objetos de interés cercanos. La siguiente operación en realizarse es erode la cual realiza una erosión o recorte del objeto con el fin de hacerlo más pequeño y asegurar su independencia.

Como último paso se emplea el método Canny que es un algoritmo que optimiza la detección de bordes con una taza muy pequeña de error, este algoritmo suprime los bordes débiles y no conectados, con esto se realiza la identificación única y localización del objeto de interés. En el código 2.5 se puede observar la implementación.

```
    Mat cierreElemento = Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(5, 5), new Point(1, 1));
    Mat kernel1 = Imgproc.getStructuringElement(Imgproc.MORPH_ERODE + Imgproc.MORPH_RECT, new Size(10, 14));
    Mat kernel = Imgproc.getStructuringElement(Imgproc.MORPH_DILATE, new Size(5, 5), new Point(1, 1));

    @Override
    public Mat process(Mat imagenEntrada) {
        Imgproc.dilate(imagenEntrada, dst, kernel);
        Imgproc.morphologyEx(dst, dst1, Imgproc.MORPH_CLOSE, cierreElemento);
        Imgproc.erode(dst1, dst2, kernel1);
        Core.bitwise_not(dst2, dst3);
        Imgproc.erode(dst3, dst4, Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(11, 10)));
        Imgproc.Canny(dst4, dst5, 2, 2);
        return dst5;
    }
}
```

Código 2.5 Clase morfología

En la Figura 2.42 se observa el resultado obtenido en esta fase.

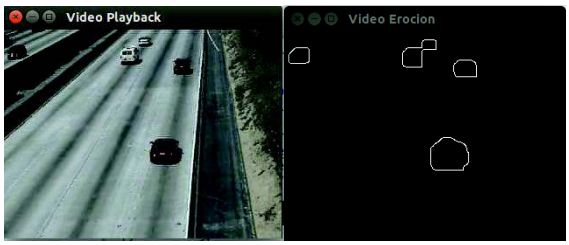


Figura 2.42 Visualización operaciones morfológicas

2.4.3.2.5 Detección y conteo

El método de Opencv que brinda soporte para la detección de contornos es findContours perteneciente a la clase Imgproc, se describe a continuación.

- **findContours. –** Una vez que se obtiene el conjunto de componentes conectados u objetos de interés definidos en la etapa anterior se hace uso de findContours para extraer los contornos de nuestros objetos de interés. A continuación se describe los parámetros que recibe esta función.
    - findContours(Mat image, java.util.List<MatOfPoint> contours, Mat hierarchy, int mode, int method)
        - **Image. –** Imagen de entrada que fue tratada previamente con umbralización o el método de Canny.
        - **Contours. –** Es una lista que almacena los putos de los contornos detectados.
        - **Hierarchy. –** Es un vector que se estableció para cada uno de los contornos hallados y sus jerarquías entre estos.
        - **Mode. –** Es un parámetro que se ocupa de las relaciones jerárquicas.
        - **Method. –** Este parámetro controla la forma en la que se aproxima los valores.

Además, como función de apoyo se tiene al método drawContorus que como su nombre lo indica permite dibujar los contornos sobre la pantalla. La clase detección se diferencia de las anteriores ya que no hace uso de la interfaz process. Se utiliza una variante modificada, debido a que el método recibe como parámetros el video procesado anteriormente y el video original. Con el primero realiza la detección aplicando su algoritmo y el con el video original se muestra los resultados de esta detección. . En el Código 2.6 se puede observar la implementación.

```
Imgproc.findContours(imagenEntrada, contornos, jerarquia, Imgproc.RETR_LIST, Imgproc.CHAIN_APPROX_TC89_KCOS);
```

Código 2.6 Método findContours

Realizada la detección de contorno se dibuja el contorno y un rectángulo sobre cada uno de los objetos de interés detectados. Para esto se ocupa el método

drawContours y *boundingRect*, sobre el rectángulo se determina el centro del mismo realizando operaciones matemáticas para determinar su posición en pixeles. En el Código 2.7 se puede observar la implementación.

```
Imgproc.drawContours(frame, contornos, idx, new Scalar(0, 0, 255));
rectanguloEnvolvente = Imgproc.boundingRect(contornos.get(idx));
Point centro = new Point((rectanguloEnvolvente.x + (rectanguloEnvolvente.width / 2)),
       (rectanguloEnvolvente.y + (rectanguloEnvolvente.height / 2)));
```

Código 2.7 Sección de código, dibujo de contornos y rectángulo

Para el conteo vehicular se traza una línea de conteo. Una vez identificados los objetos por medio de las fases anteriores, se determinó el centro de dichos objetos. Cada vez que el centro de un objeto pasa por la línea de conteo, se incrementa en uno el flujo vehicular. En el Código 2.8 se puede observar la implementación.

```
if (centro.x > 1 && centro.x <= 300 && centro.y > 100 && centro.y <= 108) {
    int cuenta = conteo++;
    Imgproc.putText(frame, String.format("cuenta: " + cuenta), new Point(20, 200), 2, 1.0,
            new Scalar(0, 0, 0, 255), 1);
    System.out.println("Se ha encontrado un carro :" + cuenta);
    LocalDate date = LocalDate.now();
    LocalTime time = LocalTime.now(ZoneId.systemDefault());
    DateTimeFormatter zonaHoraria = DateTimeFormatter.ofPattern("HH:mm:ss");
    System.out.println(time.MAX.now().format(zonaHoraria));
    System.out.println(date.toString());
    if (!ClienteOpencvUtil.guardarConteo(String.valueOf(cuenta), String.valueOf(date.now()),
            String.valueOf(time.MIN.now().format(zonaHoraria)), "2")) {
        throw new RuntimeException("No se pudo generar el registro");
    }
    Imgproc.line(frame, new Point(1, 120), new Point(300, 120), new Scalar(255, 0, 255), 6);
}
```

Código 2.8 Sección de código, conteo vehicular

Además de realizar el conteo, esta información se almacena en la BDD junto con la hora, fecha y el id de cámara. A continuación se puede observar el resultado de esta sección en la Figura 2.43.

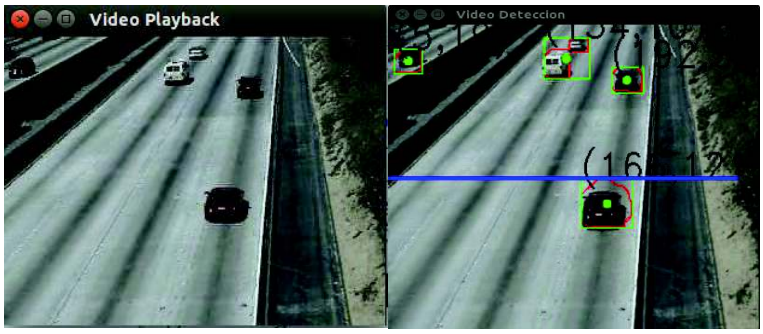


Figura 2.43 Visualización detección de contornos

2.4.4 MÓDULO DE ALMACENAMIENTO

2.4.4.1 Tecnologías utilizadas

Para el desarrollo de este módulo se optó por las siguientes tecnologías.

Postgres. – Se escogió postgres por ser un sistema de base de datos objeto relacional, la cual contiene una interfaz nativa para Java y otros lenguajes de programación, la versión utilizada fue la 9.5.2. Está disponible en [28].

SQL Power Architec. – La herramienta escogida para el modelado de datos y perfiles fue SQL Power Architec en su versión 1.0.7. Esta herramienta además del modelado de datos permite la conexión con diversos motores de base de datos. Se encuentra disponible en [29].

pgAdmin3. – Para facilitar la administración de la base de datos se hace uso del gestor gráfico pgAdmin que forma parte de las herramientas de postgreSQL, en su versión 1., el cual está disponible en [30].

2.4.4.2 Base de datos “opencvdb”

Como se observó en el apartado 2.2.5 correspondiente al diseño de este módulo, la base de datos cuenta con dos esquemas “seguridades” y “opencvapp”. Para la creación de los esquemas completos se utilizó la herramienta Power Architech, el script generado se encuentra adjunto en el ANEXO C.2.

En el Código 2.9 se puede observar la implementación.

```
CREATE DATABASE opencvdb
  WITH OWNER = reinaldo316
       ENCODING = 'UTF8'
       TABLESPACE = pg_default
       LC_COLLATE = 'es_EC.UTF-8'
       LC_CTYPE = 'es_EC.UTF-8'
       CONNECTION LIMIT = -1;

  CREATE SCHEMA opencvapp
    AUTHORIZATION reinaldo316;

  CREATE SCHEMA seguridades
    AUTHORIZATION reinaldo316;
```

Código 2.9 Creación de esquemas de BDD

2.4.5 MÓDULO DE REPORTES

2.4.5.1 Tecnologías utilizadas

De igual forma que el módulo de recepción y procesamiento, este módulo ocupa las tecnologías de Maven y Archiva en su implementación.

Java Server Faces. – Para la implementación de este módulo se optó por la utilización del framework JSF que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE, la versión escogida fue la 2.2 que es la última disponible al momento en [31].

Prime Faces. – Para la personalización del aplicativo se decidió utilizar la librería de componentes enriquecidos para JSF llamada PrimeFaces, ya que presenta un gran soporte, además de plantillas y demos para sus usuarios.

La documentación y ejemplos se encuentran disponibles en [32].

Java EE. – Todo el desarrollo del aplicativo fue llevado a cabo sobre la plataforma Java EE que utiliza un modelo de programación simplificado.

Se seguirán las recomendaciones hechas en la versión Java EE 6 y 7disponibles en [10].

GlassFish. – Como servidor de aplicación se escogió a GlassFish server en su versión 4.1 Open Source. La documentación y paquete de software se encuentran disponibles en [33].

2.4.5.2 Implementación EAR

La implementación del aplicativo Web, se realizó con el formato de empaquetado EAR (Enterprise Archive).

El cual permite crear un proyecto que contenga varios Módulos Java EE, que se desarrollan separados, pero al momento de compilar se empaquetan en uno solo [34].

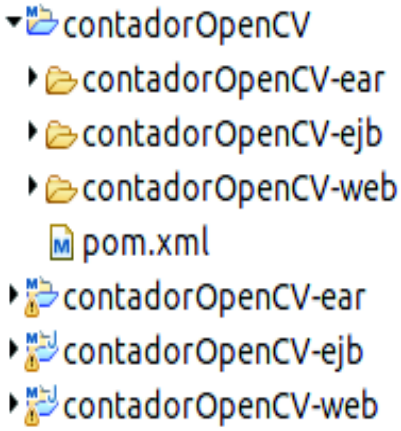


Figura 2.44 Directorio del proyecto

Como se observa en la Figura 2.44 se encuentra el directorio del proyecto llamado contadorOpenCV que tiene el formato EAR, dentro de este se tiene tres módulos Java EE.

ContadorOpenCV-ear. – Es un módulo java EE adaptador de recursos, el cual viene a ser una especie de librería para los demás módulos empaquetados, ya que contendrá JAR´s con clases ocupadas por estos.

ContadorOpenCV-ejb. – Es un módulo java EE EJB el cual contiene las clases para los Enterprise beans.

ContadorOpenCV-web. – Es un módulo java EE Web el cual contiene las clases, archivos web, xhtml, etc. El mismo que es empaquetado en un archivo .WAR el cual contiene la aplicación web completa.

2.4.5.3 Arquitectura del aplicativo web

Este módulo está basado en la arquitectura Java EE, la cual propone 3 capas que son las siguientes:

- **Capa de presentación. –** Encargada de la interfaz gráfica.
- **Capa de negocio. –** Encargada del manejo de toda la lógica del negocio.
- **Capa de acceso a datos. –** Encargada de todas las operaciones realizadas con la base de datos.

2.4.5.4 Implementación capa acceso a datos y negocio

Para la implementación de la capa de acceso a datos se utilizó JPA (Java Persistance API).

Para gestionar la persistencia con la base de datos de los registros generados desde el aplicativo. Y EJB (Enterprise Java Beans) para la gestión de la transaccionalidad de los datos según la guía ofrecida en [10].

Se hizo uso del patrón de diseño de persistencia DAO (Data Access Object) el cual sugiere dividir las responsabilidades en el aplicativo, de tal forma que se tenga unas clases que se encargaran de la lógica de negocio y otras clases la responsabilidad de persistencia [35].

En la Figura 2.45 se puede observar el diagrama de clases implementado.

Como se observa en la parte más baja del diagrama se encuentra la capa de acceso a datos.

Aquí se encuentran las entidades mapeadas desde la base de datos, en el siguiente nivel se encuentra sus correspondientes clases DAO que heredan todos los métodos implementados en la clase DAO genérico.

El siguiente nivel pertenece a la capa de negocio, en la cual se puede observar las clases Puente que sirven como pasarela entre las clases servicio que son EJB´s y sus respectivas clases Dao.

Las clases servicio serán las que ofrecerán todos los métodos de la capa de acceso a datos a la capa web. Con esto se está optimizando el patrón de diseño Fachada el cual tiene como característica el brindar solo una puerta de entrada a otros subsistemas, en este caso las clases servicio dan acceso a los métodos contenidos en la capa de negocio y acceso a datos.

En la Figura 2.45 se puede observar el diagrama de clases implementado, el cual consta de las dos capas antes mencionadas.

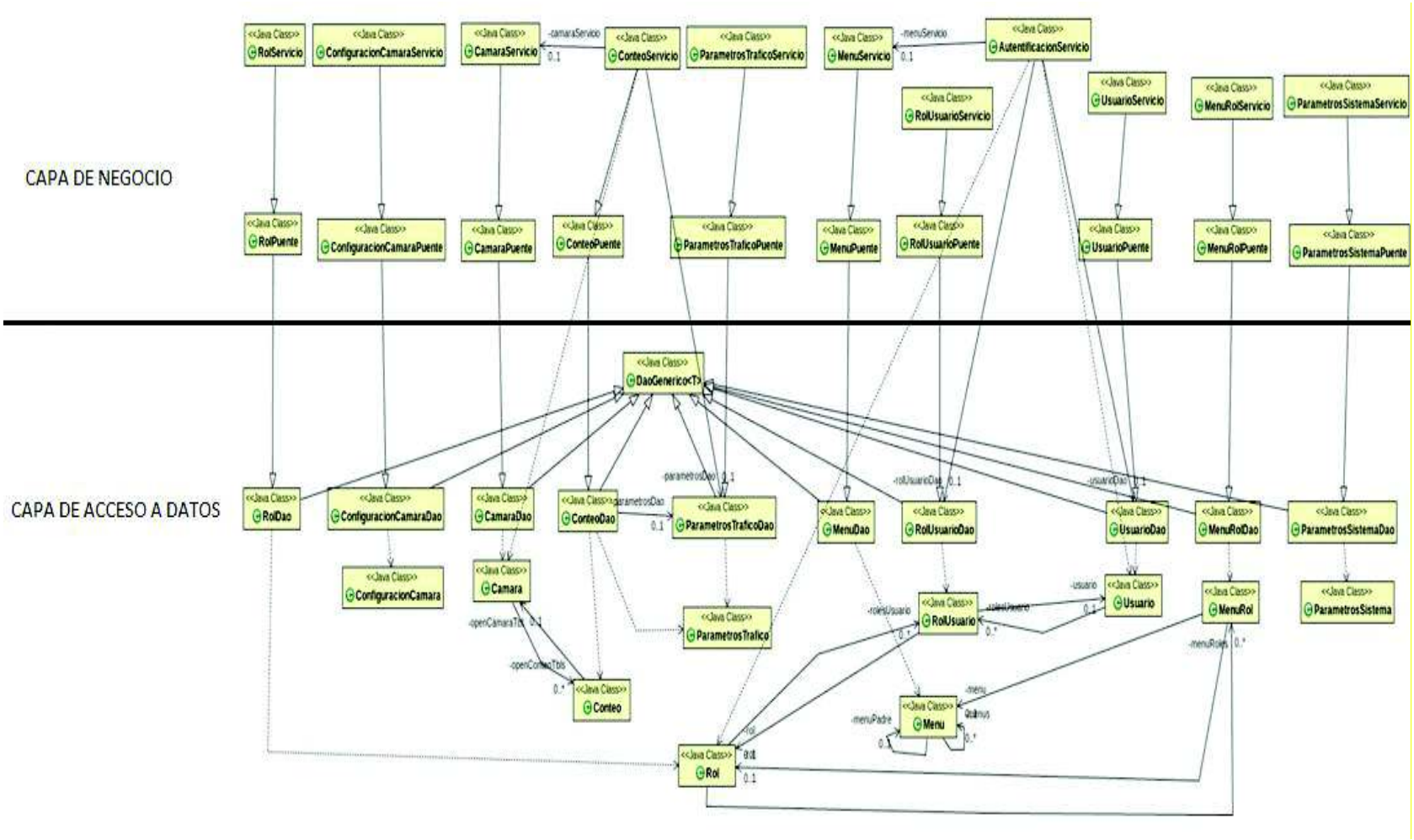


Figura 2.45 Diagrama de clases de la capa de acceso a datos y negocio

2.4.5.4.1 Mapeo de base de datos

La API de persistencia de Java brinda un eje de mapeo objeto - relacional para la gestión de datos en aplicaciones Java. Con esto se puede genera automáticamente una entidad por cada tabla en la base de datos. Lo primero es tener una conexión con la base de datos como se observa en la Figura 2.46.

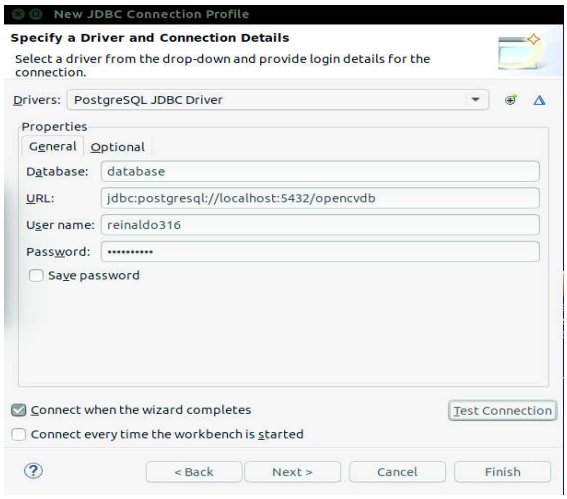


Figura 2.46 Conexión a la base de datos por JPA

Con la conexión establecida se mapea la base de datos indicando el esquema. Una vez realizada la conexión, se crea las clases que son las entidades como se observa en la Figura 2.47.

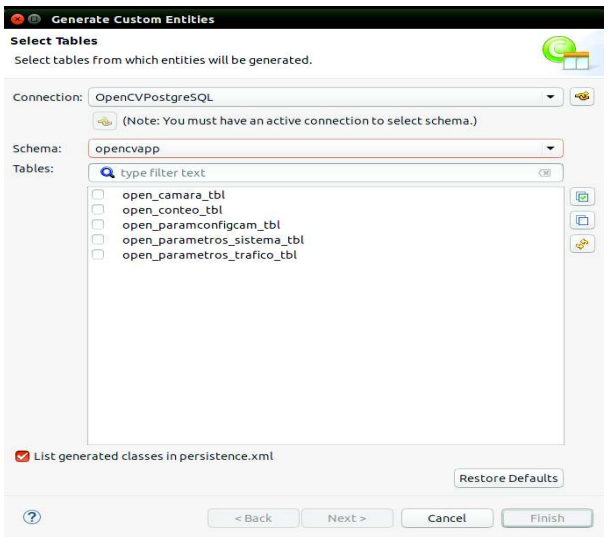


Figura 2.47 Generación de entidades desde tablas por JPA

A continuación se describe la clase Usuario a manera de ejemplo ya que las demás clases y sus entidades tienen un formato similar. El código completo de la implementación se encuentra en el ANEXO C.3 En el Código 2.10 se puede observar la implementación.

```
@Entity
@Table(name = "seg_usuarios_tbl", schema = "seguridades")
@NamedQuery(name = Usuario.LISTAR_TODO, query = "SELECT u FROM Usuario u where u.estado='True'")
@NamedQueries({
        @NamedQuery(name = Usuario.BUSCAR_USUARIO_USERNAME, query = "SELECT u FROM Usuario u where u.estado= 'True' and u.username= :username")
        @NamedQuery(name = Usuario.VALIDAR_LOGIN, query = "SELECT u FROM Usuario u, RolUsuario ru, Rol r where ru.usuario.id = u.id and r.id =
public class Usuario implements Serializable {

    private static final long serialVersionUID = 3149486219832163609L;
    public static final String LISTAR_TODO = "Usuario.listarTodo";
    public static final String BUSCAR_USUARIO_USERNAME = "Usuario.buscarUsuarioUsername";
    public static final String VALIDAR_LOGIN = "Usuario.login";

    @Getter
    @Setter
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "us_id", unique = true)
    private Long id;

    @Getter
    @Setter
    @Column(name = "us_apellido", nullable = false, length = 2147483647)
    private String apellido;
```

Código 2.10 Fragmento código, clase Usuario

Como se observa en la Figura 2.47 al utilizar JPA para mapear la base de datos se crea entidades por cada tabla. La entidad Usuario, tiene sus propios atributos que están enlazados por la persistencia a la base de datos gracias a JPA. A continuación se describe cada anotación perteneciente a JPA:

- **@Entity. –** Indica que esta clase es una entidad, siempre se coloca al inicio de la clase.
- **@Table. –** Indica el nombre de la tabla y el esquema de base de datos
- **@NamedQuery. –** Permite realizar consultas a la base de datos con el formato SQL pero utilizando los atributos definidos en la clase.
- **@Getter y @Setter. –** Son parte de la librería Loombok e implementa de forma transparente los getters y setters de cada atributo,
- **@Id. –** Es la identificación de primary key de la entidad.

- **@GeneratedValue. –** Indica que la primary key es autogenerada y se está ocupando la secuencia definida en la base de datos mediante la declaración (strategy = GenerationType.IDENTITY).
- **@Column. –** Especifica que la columna está asociada a un atributo en la clase.

Como se puede ver en el apartado de diseño del módulo de almacenamiento también se tiene relaciones de mucho a muchos en algunas de las tablas y por tal motivo se tiene dos tablas de rompimiento. Se toma como ejemplo la relación entre la entidad Rol y Menú, la cual indica que un rol puede tener muchos menús y viceversa. Por esto se creó la tabla MenuRol en la base de datos y su correspondiente entidad. En el Código 2.11 y el Código 2.12 se puede observar la implementación.

- **@JoinColumn. –** Indica la tabla con la que se hace la relación.
- **@ManyToOne. –** Indica una relación unidireccional de muchos a uno.

```
// bi-directional many-to-one association to Menu
@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "me_id", nullable = false, insertable = false, updatable = false)
private Menu menu;
```

Código 2.11 Fragmento código, clase Menú-rol

@OneToMany. – Indica una relación unidireccional de uno a muchos

```
//bi-directional many-to-one association to MenuRol
    @Getter
    @Setter
    @OneToMany(mappedBy="rol")
    private List<MenuRol> menuRoles;
```

Código 2.12 Fragmento código, clase Rol

2.4.5.4.2 Data Access Object

Como se pude ver en la Figura 2.45 cada una de las entidades creadas tiene su propio DAO. Siguiendo el ejemplo planteado se explicará la implementación de la clase UsuarioDao ya que las demás clases DAO tendrán un formato similar, el código completo de la implementación del resto de clases DAO se encuentra en el

ANEXO C.3. En el Código 2.13 se puede observar la implementación. De la clase UsuarioDao.

```
@Stateless
@LocalBean
public class UsuarioDao extends DaoGenerico<Usuario> {
```

Código 2.13 Fragmento de código, clase UsuarioDao

A continuación se describe las anotaciones pertenecientes a EJB:

- **@Stateless. –** Indica que esta clase es un Bean de sesión sin estado correspondiente al estándar EJB. Esta notación muestra que no se mantiene un estado permanente de conversación con el cliente, solo mantendrá un estado cuando se realice una invocación de los métodos.
- **@LocalBean. –** Permite hacer uso del bean de manera local por parte de un cliente.

La clase UsuarioDao hereda todos los métodos de una clase llamada DaoGenerico, en la cual se ha implementado los métodos para realizar consultas de los datos a través de las entidades JPA. El código de la implementación de la clase DaoGenerico se encuentra en el ANEXO C.3.

2.4.5.4.3 Servicios

En la Figura 2.45, se aprecia que cada una de las entidades creadas tiene su propio DAO y estas a su vez están asociadas a una clase Servicio, esta asociación se da a través de una clase denominada Puente. La clase Servicio será la encargada de gestionar las operaciones que se soliciten desde la capa presentación.

Siguiendo el ejemplo planteado se explicará la implementación de la clase UsuarioPuente y UsuarioServicio ya que las demás clases tanto Puentes como Servicios tendrán un formato similar. El código completo de la implementación de las clases Puente y Servicio de cada entidad se encuentra en el ANEXO C.3. En el Código 2.14 se puede observar la implementación de la clase UsuarioPuente. Esta es una clase que hereda los métodos de UsuarioDao, y sirve de pasarela para llegar a la clase UsuarioServicio. Por lo cual no contiene ningún método propio.

```
public class UsuarioPuente extends UsuarioDao {
}
```

Código 2.14 Clase UsuarioPuente

En el Código 2.15 se puede observar la implementación de la clase UsuarioServicio, esta lleva las etiquetas de **@LocalBean y @Stateless** lo que convierten a esta clase en un EJB. Esta clase hereda los métodos de UsuarioPuente que a la vez hereda los métodos de la clase UsuarioDao, con esto se está optimizando el patrón Fachada que tiene como característica principal ofrecer una puerta de entrada hacia otro subsistema.

```
@LocalBean
@Stateless
public class UsuarioServicio extends UsuarioPuente {

    public String encriptarPass (String pass){

        return EncriptacionUtil.calculateHash(pass);

    }
}
```

Código 2.15 Clase UsuarioServicio

2.4.5.4.4 Comunicación Json

La comunicación entre el módulo de recepción y procesamiento y el módulo de almacenamiento requiere de una capa de servicios adicionales. Estos servicios permiten inter operar los módulos y realizar la transacción de datos mediante un EJB hacia la capa de acceso datos. La comunicación se realiza mediante servicios Json (JavaScript Object Notation) que es un formato ligero de intercambio de datos.

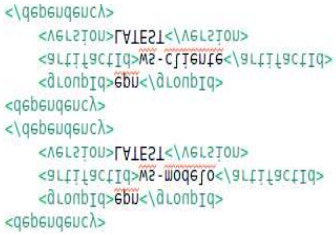


Figura 2.48 Dependencias ws-modelo y ws-cliente en archivo POM.xml

Esta capa de servicios adicionales consta de dos JAR´s llamados ws-modelo y ws-cliente, los mismos que se encuentran subidos al repositorio local de Archiva y agregados como dependencias en los archivos POM.xml tanto en la aplicación Cliente Opencv Swing del módulo de recepción y procesamiento y en el POM.xml del módulo Java EE contadorOpenCV-web perteneciente al módulo de reportes.

En la Figura 2.48 se observa las dependencias agregadas en el archivo POM de la aplicación Cliente Opencv Swing.

Ws-modelo. – Aquí se especifica el modelo de datos que se van a enviar y recibir. Este JAR es usado para convertir los datos enviados en String por Json, al tipo que se requiera para enviarlos a persistir a la BDD. Consta de 4 clases:

- **CodigoRespuestaWS,** clase donde se especifica las dos opciones posibles de respuesta al envió "OK" o "ERROR".
- **ConteoEntrada,** clase en la cual se especifica los parámetros que se recibirá en la comunicación.
- **ConteoSalida,** clase que especifica el String de confirmación.
- **ConteoRespuesta,** clase en la cual se especifica el código de respuesta, el mensaje y el conteo salida.

Ws-cliente. – Aquí se efectúa la conversión de datos y él envió al servidor. Para realizar esta conversión se utiliza la librería google-gson que permite convertir objetos Java en la representación Json y viceversa.

El código de implementación de estos módulos está adjuntos en el ANEXO C.3.

Implementación envió de conteo al servidor

Para la implementación se creó la clase ClienteOpencvUtil en el módulo de recepción y procesamiento, en la cual se hace uso del método readWs de la clase ClienteUtil del ws-cliente para realizar él envió de datos al servidor

En el Código 2.16 se puede observar la implementación.

```
public static boolean guardarConteo(String conteo, String fecha, String hora, String idCamara) {
    ConteoRespuesta respuesta = new ConteoRespuesta();
    ConteoEntrada entrada = new ConteoEntrada();
    entrada.setConteo(conteo);
    entrada.setFecha(fecha);
    entrada.setHora(hora);
    entrada.setIdCamara(idCamara);
    ClienteUtil<ConteoRespuesta> cliente = new ClienteUtil<ConteoRespuesta>();
    respuesta = cliente.readWs(Constante.URL_SERVICIO, entrada, respuesta);
    if (respuesta != null && respuesta.getCodigo().equals(CodigoRespuestaWs.OK.getCodigo()))
        return true;
    return false;
}
```

Código 2.16 Clase ClienteOpencvUtil

Implementación del almacenamiento de conteo en la BDD

Para la implementación se creó la clase ClienteServicioRest en el módulo contadorOpenCV-web, en la cual se consume los métodos ofrecidos por el EJB ConteoServicio. Se creó el método guardarConteo que recibe el objeto ConteoEntrada, el cual contiene los parámetros enviados desde la aplicación cliente. A estos parámetros se les convierte al tipo de dato que se necesite para poder guardarlos mediante el servicio EJB. En el Código 2.17 se puede observar la implementación

```
public ConteoRespuesta guardarConteo(ConteoEntrada registroEntrada) {
    Conteo conteo = new Conteo();
    conteo.setContFecha(parseDate(registroEntrada.getFecha()));
    conteo.setContHora(parseHour(registroEntrada.getHora()));
    conteo.setContVehicular(Long.parseLong(registroEntrada.getConteo()));
    ConteoPK pk = new ConteoPK();
    pk.setCamId(Long.parseLong(registroEntrada.getIdCamara()));
    conteo.setId(pk);
    ConteoRespuesta respuesta = new ConteoRespuesta();
    try {
        servicio.guardar(conteo);
        respuesta.setCodigo(CodigoRespuestaWs.OK.getCodigo());
        ConteoSalida salida = new ConteoSalida();
        salida.setConfirmacion("Registro exitoso");
        respuesta.setConteo(salida);
    } catch (ServicioExcepcion ex) {
        log.log(Level.SEVERE, null, ex);
        respuesta.setCodigo(CodigoRespuestaWs.ERROR.getCodigo());
        respuesta.setMensaje(ex.getMessage());
    }
    return respuesta;

}
```

Código 2.17 Método guardarConteo.

2.4.5.5 Implementación de la capa Web

Teniendo en cuenta lo que sugiere el estándar EJB, el diagrama de clases muestra un contenedor web que será el que interactúe con la capa de presentación (clases terminadas en bean).

Por otro lado un contenedor de aplicación interactuará con la capa de acceso a datos (clases terminadas en controlador). En la Figura 2.49 se muestra el diagrama de clases para la capa negocio.

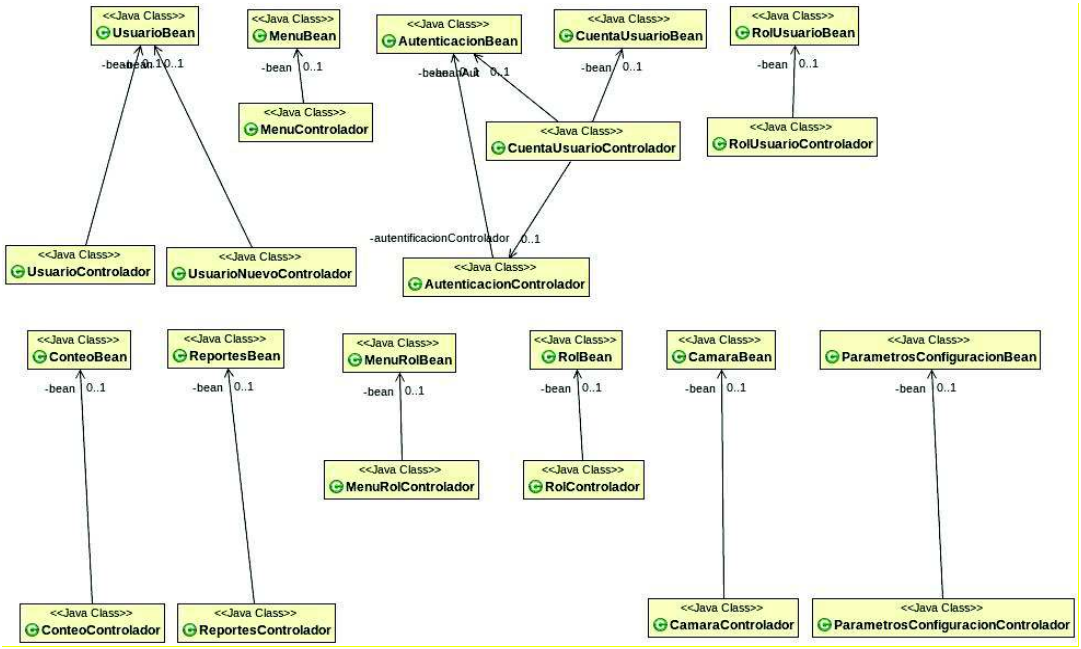


Figura 2.49 Diagrama de clases, capa de presentación

Para la implementación de esta sección se hace uso del patrón MVC para facilitar el desarrollo e implementación. A continuación se explicará la implementación la gestión de datos. El código de implementación de este módulo de reportes esta adjunto en el ANEXO C.3.

2.4.5.5.1 Gestión

El aplicativo presenta algunos módulos de gestión entre los que se encuentran la gestión de usuarios, menús, roles, cámaras, relación menú-rol y relación usuario-rol. Estos módulos cumplen con las operaciones CRUD para sus respectivos datos, y el proceso que se detalla en cada uno es similar.

Por lo cual a manera de ejemplo se presenta la implementación de la gestión de usuarios correspondiente al rol de administrador.

En el ANEXO C.4 se puede encontrar la implementación de los módulos más representativos de la capa Web.

El módulo de getion de usuarios ofrece al administrador una interfaz en la cual se realizara las siguientes acciones:

- Ver el listado de usuarios registrados.
- Crear un nuevo usuario.
- Actualizar un usuario.
- Eliminar un usuario.

En la Figura 2.50 se puede apreciar el funcionamiento MVC aplicado a la gestión de usuarios.

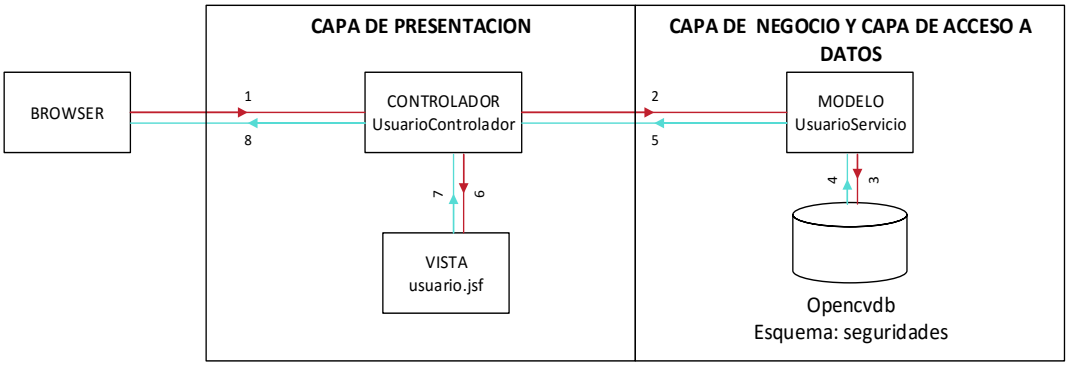


Figura 2.50 MVC para gestión de usuarios

Modelo

El modelo es el encargado de gestionar, procesar y validar datos, todas estas funcionalidades fueron abordadas en el apartado 2.3.5.3. De acuerdo al patrón de diseño Fachada, el servicio EJB que brindará acceso a las funcionalidades, en este caso específico será UsuarioServicio mediante el cual se tendrá acceso a los siguientes métodos.

- **Guardar. –** Método que me permite guardar un parámetro en la base de datos, este método es heredado de la clase DaoGenerico.

- **listarUsuario. –** Método que me devuelve una lista de usuarios, con todas sus dependencias mientras su estado sea verdadero.

Vista

Es la encargada de la interacción con el usuario, se utilizó la plantilla Adamantium de PrimeFaces cuyo demo está disponible en [28]. Dicha plantilla está basada en

JSF, HTML5 y CSS3, con lo cual se facilitó la implementación de las diferentes vistas.

En Figura 2.51 y la Figura 2.52 se aprecia la vista implementada. Llamada usuario.jsf.

Figura 2.51 Sección de visualización de usuarios de la vista usuario.jsf

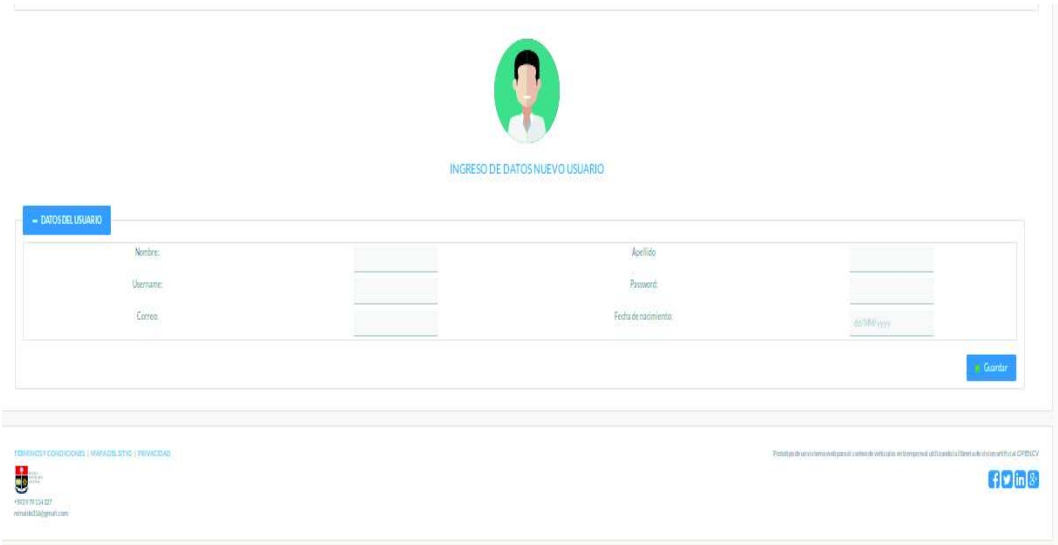


Figura 2.52 Sección de operaciones CRUD de la vista usuario.jsf

Controlador

Es el encargado de relacionar la vista y el modelo. La clase que tiene a cargo las funcionalidades de controlador es UsuarioControlador y la clase UsuarioBean que se utiliza para instanciar las variables ocupadas en la vista.

Los métodos utilizados en este controlador son los siguientes.

- **seleccionarUsuario. –** Método mediante el cual se carga el objeto Usuario con los parámetros ingresados por el administrador, En el Código 2.18 se puede observar la implementación

```
public void seleccionarUsuario(Usuario s) {
    getBean().setUsuarioSeleccionado(s);
    System.out.println(s);
    // RequestContext.getCurrentInstance().update(":frmUsuario");
}
```

Código 2.18 Método seleccionarUsuario

- **cargarUsuario. –** Método mediante el cual se solicita el listado de usuarios con sus respectivos parámetros, que es cargado en un array list, En el Código 2.19 se puede observar la implementación

```
private void cargarUsuarios() {
    try {
        getBean().iniciar();
        getBean().setUsuarios(getUsuarioServicio().listarUsuarios());
    } catch (ServicioExcepcion e) {
        log.log(Level.FINE, e.getMessage(), e);
        ponerMensajeError(ConstanteUtil.MENSAJE_ERROR_INESPERADO);
    }
```

Código 2.19 Método cargarUsuario

- **nuevoUsuario. –** Método mediante el cual reinicio los parámetros de mi objeto usuario seleccionado En el Código 2.20 se puede observar la implementación

```
public void nuevoUsuario() {
    getBean().setUsuarioSeleccionado(new Usuario());

}
```

Código 2.20 Método nuevoUsuario

- **validarDatosUsuario. –** Método mediante el cual se realiza una validación de formato de los parámetros ingresados por el administrador, En el Código 2.21 se puede observar la implementación

```
public boolean validarDatosUsuario() {
    Usuario usu = getBean().getUsuarioSeleccionado();
    boolean resultado = true;
    if (usu.getNombre() == null || usu.getNombre().isEmpty()) {
        resultado = false;
        ponerMensajeError("Ingrese el nombre.");
    }

    if (usu.getApellido() == null || usu.getApellido().isEmpty()) {
        resultado = false;
        ponerMensajeError("Ingrese el apellido.");
    }

    if (usu.getCorreo() == null || usu.getCorreo().isEmpty()) {
        resultado = false;
        ponerMensajeError("Ingrese el correo.");
    } else {
        if (!validarCorreo(usu.getCorreo())) {
            ponerMensajeError("Ingrese un correo válido.");
            resultado = false;
        }
    }

    if (usu.getFechaNacimineto() == null) {
        resultado = false;
        ponerMensajeError("Ingrese la fecha de nacimiento.");
    }

    if (usu.getUsername() == null || usu.getUsername().isEmpty()) {
        resultado = false;
        ponerMensajeError("Ingrese el username.");
    }
```

Código 2.21 Método validarDatosUsuario

- **crearActualizarUsuario. –** Método mediante el cual se realiza las operaciones CRUD, dependiendo del requerimiento del administrador, En el Código 2.22 se puede observar la implementación

```
public void crearActualizarUsuario(Usuario s, boolean eliminar) {
    getBean().setUsuarioSeleccionado(s);
    if (validarDatosUsuario()) {
        if (eliminar)
            s.setEstado(Boolean.FALSE);
        if (s.getId() == null) {
            s.setEstado(Boolean.TRUE);
            //s.setPassword("password");
        }

        try {
            s.setPassword(getUsuarioServicio().encriptarPass(s.getPassword()));
            getUsuarioServicio().guardar(s);
            ponerMensajeInfo(ConstanteUtil.MENSAJE_REGISTRO_EXITOSO);
            cargarUsuarios();
        } catch (ServicioExcepcion e) {

            log.log(Level.FINE, e.getMessage(), e);
            ponerMensajeError(ConstanteUtil.MENSAJE_ERROR_INESPERADO);
        }
    } else {
        ponerMensajeError("Por favor valide los datos del usuario a guardar");
    }
}
```

Código 2.22 Método crearActualizarUsuario

Un proceso similar se realizó con los demás módulos de gestión, el código completo de las implementaciones se encuentra en el ANEXO C.3.

CAPÍTULO 3: PRUEBAS

En este capítulo se realiza una descripción del escenario de pruebas empleado en la validación del sistema. También se presentarán los resultados de las pruebas realizadas a cada módulo y del sistema completo.

3.1 ESCENARIO DE PRUEBA

El escenario desplegado para las pruebas fue un ambiente controlado en el cual el usuario tenga acceso a la red en la que se encuentra el servidor para que pueda ingresar al prototipo a través de una computadora personal y/o un teléfono móvil. Tal y como se observa en la Figura 3.1.

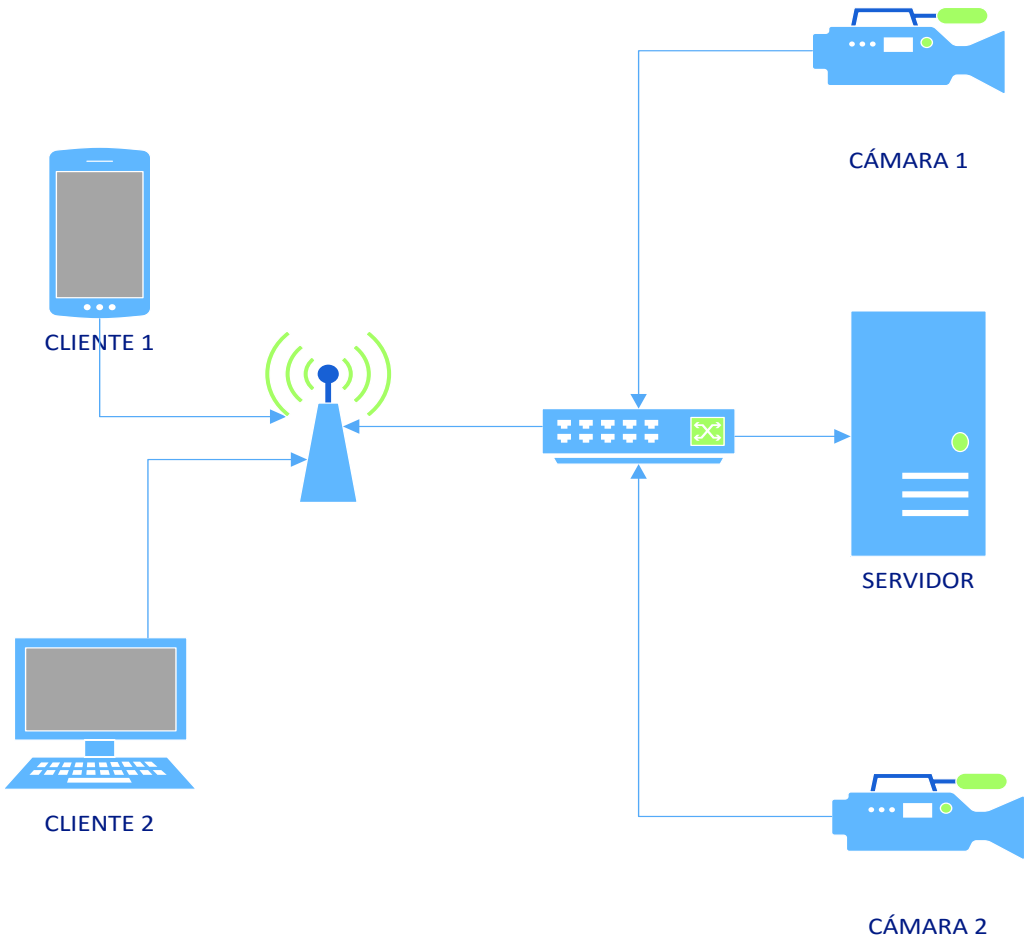


Figura 3.1 Escenario de pruebas

En la Tabla 3.1 se puede observar las características de hardware y software de los dispositivos utilizados en el escenario de pruebas además de las cámaras cuyas características se encuentran descritas en la Tabla 2.32 y la Tabla 2.33.

Tabla 3.1 Características de los dispositivos

Dispositivos	**Características**	
	Hardware	**software**
Servidor	Marca: HP Procesador: Core i 7 Memoria RAM: 8 Gb	Sistema operativo: Ubuntu Server 15.1
Computador	Marca: Acer Procesador: Core i 5 Memoria RAM: 6 Gb	Sistema operativo: Windows 10 Navegador: Chrome
Teléfono inteligente	Marca. Huawei Modelo: P8 Procesador: Kirin 930 de 64 bits, ocho núcleos a 2GHz Memoria RAM: 3 Gb	Sistema operativo: Android 5.1 lollipop Navegador: Chrome

3.2 PRUEBAS DE FUNCIONALIDAD

Se realizaron las pruebas de funcionalidad considerando cada una de las historias de usuario y sus respetivos criterios de aceptación, con el objetivo de verificar la satisfacción del usuario con respecto al funcionamiento esperado.

Las pruebas de funcionalidad están basadas en los roles de usuario y sus respectivas historias de usuario con sus criterios de aceptación, en las siguientes secciones se muestra las pruebas realizadas.

3.2.1 FUNCIONALIDADES DE ADMINISTRADOR

Las pruebas de funcionalidad de administrador cubren 5 historias de usuario con 5 criterios de aceptación cada una.

A manera ejemplo se explica cómo se realizó las pruebas de funcionalidad a la primera historia de usuario y sus respectivas historias de usuario

3.2.1.1 HUA001 - Gestionar usuarios

Se especificaron 3 pruebas para comprobar los criterios de aceptación de la historia, estas se describen a continuación.

3.2.1.1.1 Crear usuarios

Esta prueba se realizó para comprobar los dos primeros criterios de aceptación de la historia de usuario.

Se crea un nuevo usuario. – En caso de ingresar correctamente los campos solicitados, sale un aviso de registro exitoso y se actualiza la tabla de visualización como se observa en la Figura 3.2.

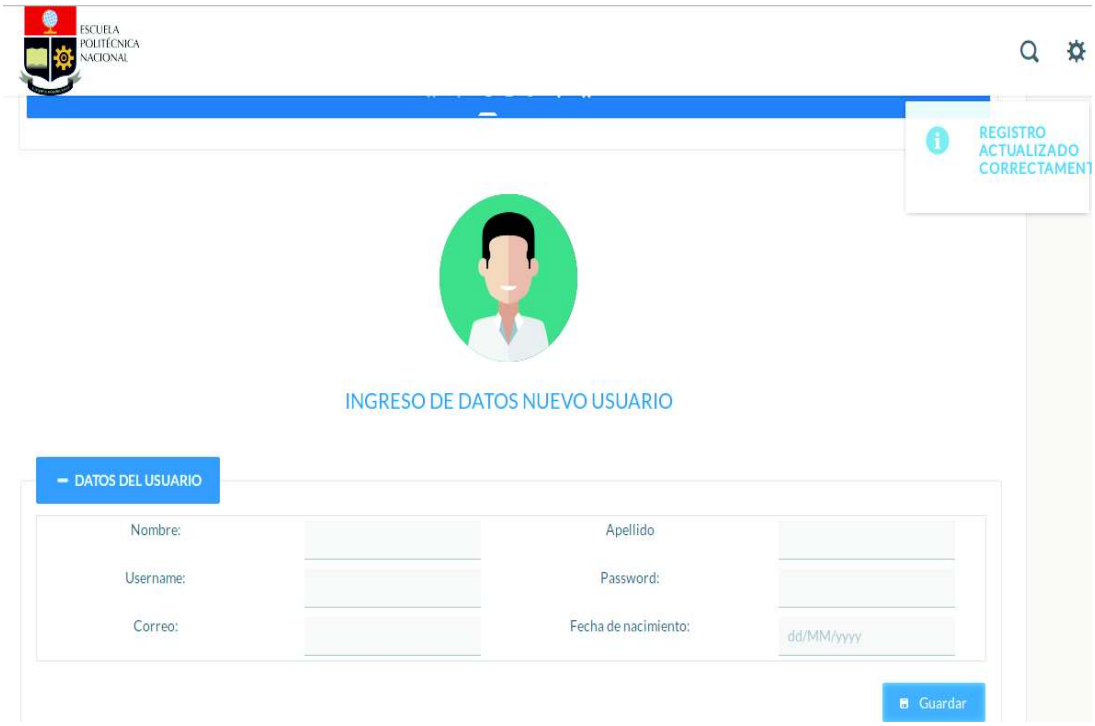


Figura 3.2 Captura vista creación de usuario

No se crea un nuevo usuario. – En caso de no ingresar correctamente los campos solicitados, sale un aviso de fallo como se observa en la Figura 3.3.

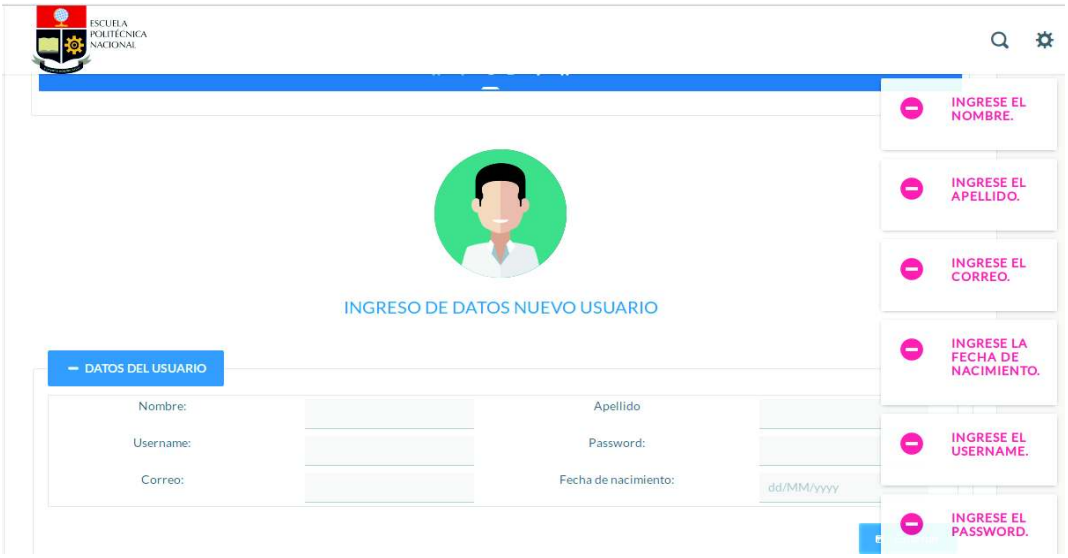


Figura 3.3 Captura vista de la no creación de usuario

3.2.1.1.2 Actualizar usuario

Esta prueba se realizó para comprobar el tercer y cuarto criterio de aceptación de la historia de usuario.

Se actualiza un usuario. – En caso de ingresar correctamente los campos solicitados, sale un aviso de registro exitoso y se actualiza la tabla de visualización como se observa en la Figura 3.4.

Figura 3.4 Captura de la vista de actualizar usuarios

No se actualiza un usuario. – En caso de no ingresar correctamente los campos solicitados, sale un aviso de fallo como se observa en la Figura 3.5.

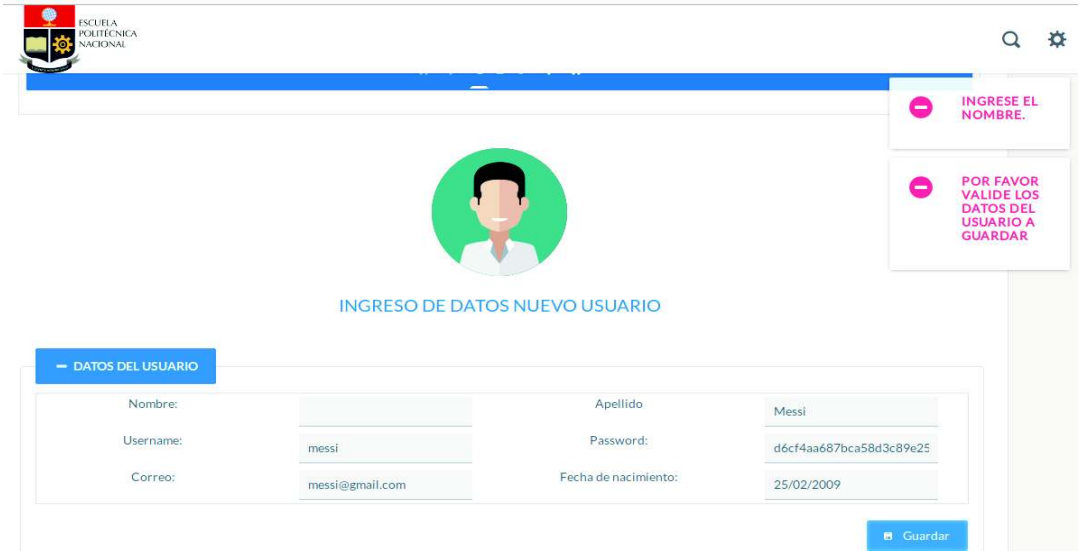


Figura 3.5 Captura de la no actualización de usuarios

3.2.1.1.3 Eliminar usuario

Esta prueba se realizó para comprobar el último criterio de aceptación de la historia de usuario.

Se elimina un usuario. – En caso de seleccionar correctamente al usuario, sale un aviso de registro exitoso y se actualiza la tabla de visualización como se observa en la Figura 3.6.

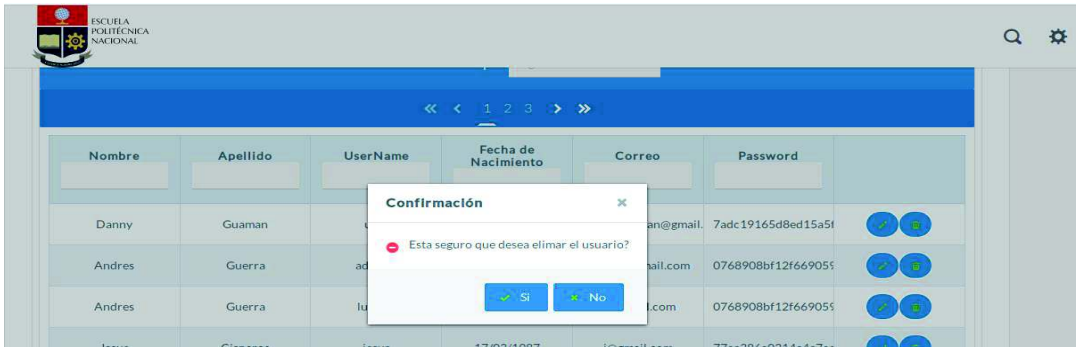


Figura 3.6 Captura de la vista de eliminación de usuarios

De igual manera se realizó la comprobación del resto de historias de usuario y sus criterios de aceptación para el rol administrador, en la Tabla 3.2 se puede observar todos los resultados de las pruebas de aceptación, pertenecientes al rol administrador.

Tabla 3.2 Tabla de resultados

Historia de usuario	criterio de aceptación	laptop	teléfono inteligente	observación
HUA001	Se crea un usuario	OK	OK	Para mejor visualización en el teléfono móvil la pantalla debe estar de forma horizontal
	No se crea un usuario	OK	OK	
	Se actualiza un usuario	OK	OK	
	No se actualiza un usuario	OK	OK	
	Se elimina un usuario	OK	OK	
HUA002	Se crea un rol de usuario	OK	OK	Para mejor visualización en el teléfono móvil la pantalla debe estar de forma horizontal
	No se crea un rol de usuario	OK	OK	
	Se actualiza un rol de usuario	OK	OK	
	No se actualiza un rol de usuario	OK	OK	
	Se elimina un rol de usuario	OK	OK	
HUA003	Se crea un menú de usuario	OK	OK	Para mejor visualización en el teléfono móvil la pantalla debe estar de forma horizontal
	No se crea un menú de usuario	OK	OK	
	Se actualiza un menú de usuario	OK	OK	
	No se actualiza un menú de usuario	OK	OK	
	Se elimina un menú de usuario	OK	OK	
HUA004	Se crea una relación usuario- menú	OK	OK	Para mejor visualización en el teléfono móvil la pantalla debe estar de forma horizontal
	No se crea una relación usuario-menú	OK	OK	
	Se actualiza una relación usuario- menú	OK	OK	
	No se actualiza una relación usuario- menú	OK	OK	
	Se elimina un una relación usuario- menú	OK	OK	
HUA005	Se crea una relación usuario-rol	OK	OK	Para mejor visualización en el teléfono móvil la pantalla debe estar de forma horizontal
	No se crea una relación usuario-rol	OK	OK	
	Se actualiza una relación usuario-rol	OK	OK	
	No se actualiza una relación usuario-rol	OK	OK	
	Se elimina un una relación usuario-rol	OK	OK	

3.2.2 FUNCIONALIDADES DE OPERADOR

Las pruebas de funcionalidad de operador cubren 5 historias de usuario, una con 5 criterios de aceptación y las otras cuatro con 2.

3.2.2.1 HUO001 - Gestionar cámaras

Se especificaron 3 pruebas para comprobar los criterios de aceptación de la historia, estos se describen a continuación.

3.2.2.1.1 Crear cámara

Esta prueba se realizó para comprobar los dos primeros criterios de aceptación de la historia de usuario.

Se crea una nueva cámara. – En caso de ingresar correctamente los campos solicitados, sale un aviso de registro exitoso y se actualiza la tabla de visualización como se observa en la Figura 3.7.

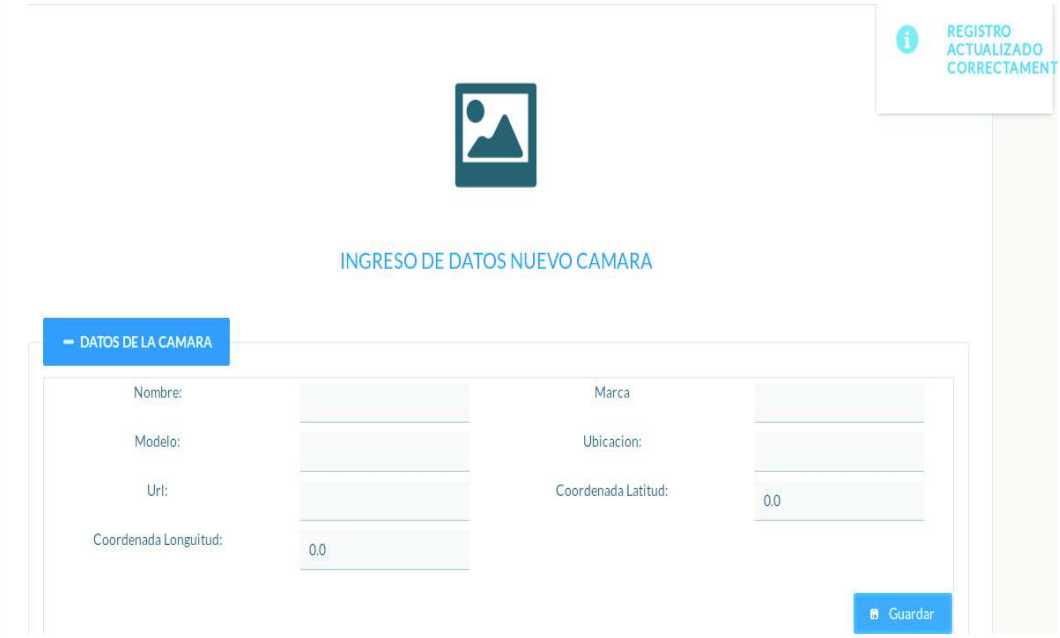


Figura 3.7 Captura de vista de nueva cámara

No se crea una nueva cámara. – En caso de no ingresar correctamente los campos solicitados, sale un aviso de fallo como se observa en la Figura 3.8.

Figura 3.8 Captura de Vista de no guardado de información de la cámara

3.2.2.1.2 Actualizar cámara

Esta prueba se realizó para comprobar el tercer y cuarto criterios de aceptación de la historia de usuario.

Se actualiza una cámara. – En caso de ingresar correctamente los campos solicitados, sale un aviso de registro exitoso y se actualiza la tabla de visualización similar al que se observa en la Figura 3.9.

Figura 3.9 Captura vista de actualización de cámara

No se actualiza una cámara. – En caso de no ingresar correctamente los campos solicitados, y como en el caso de la no creación de cámaras sale un aviso de fallo similar al que se observa en la Figura 3.8.

3.2.2.1.3 Eliminar cámara

Esta prueba se realizó para comprobar el último criterio de aceptación de la historia de usuario.

Se elimina una cámara. – En caso de seleccionar correctamente la misma, sale un aviso de registro exitoso y se actualiza la tabla de visualización como se observa en la Figura 3.10.

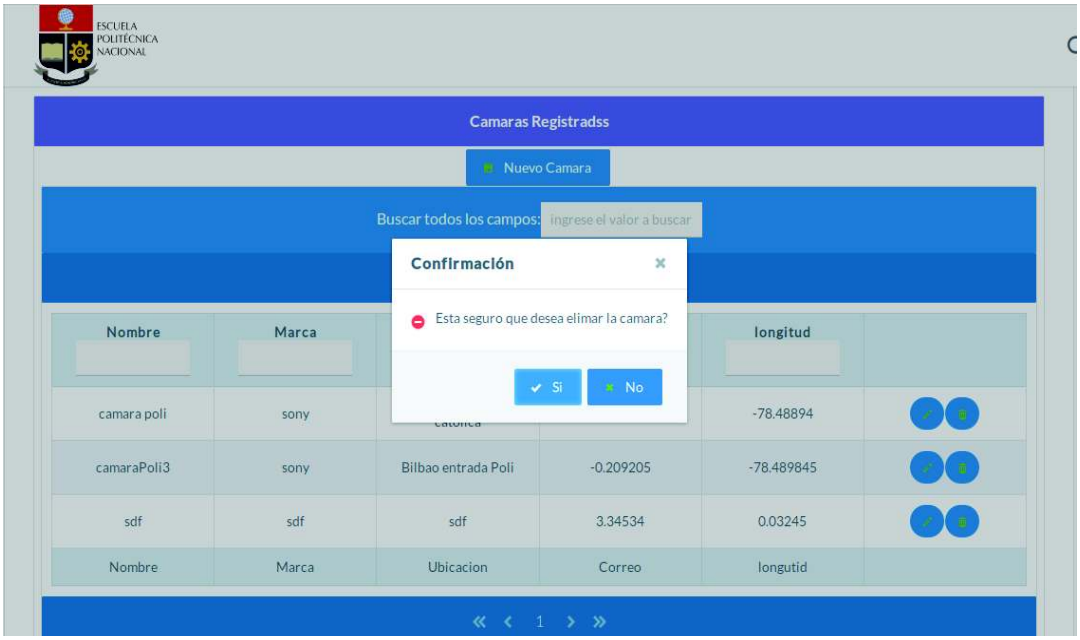


Figura 3.10 Captura vista de la eliminación de una cámara

3.2.2.2 HUO002 -Modificar parámetros

Esta historia de usuario consta de dos criterios de aceptación que se describen a continuación.

Se actualiza los parámetros. – En caso de ingresar correctamente los campos solicitados, sale un aviso de registro exitoso y se actualiza la tabla de visualización como se observa en la Figura 3.11.

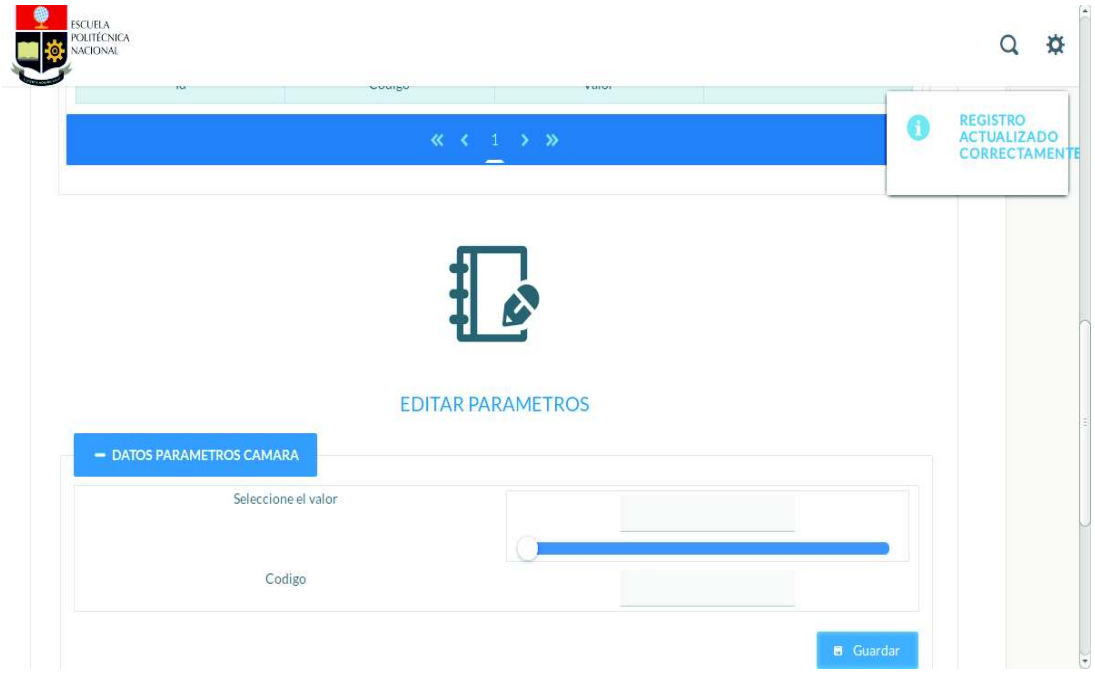


Figura 3.11 Captura de vista de modificación de parámetros

No se actualiza los parámetros. – En caso de no ingresar correctamente los campos solicitados, sale un aviso de fallo como se observa en la Figura 3.12.

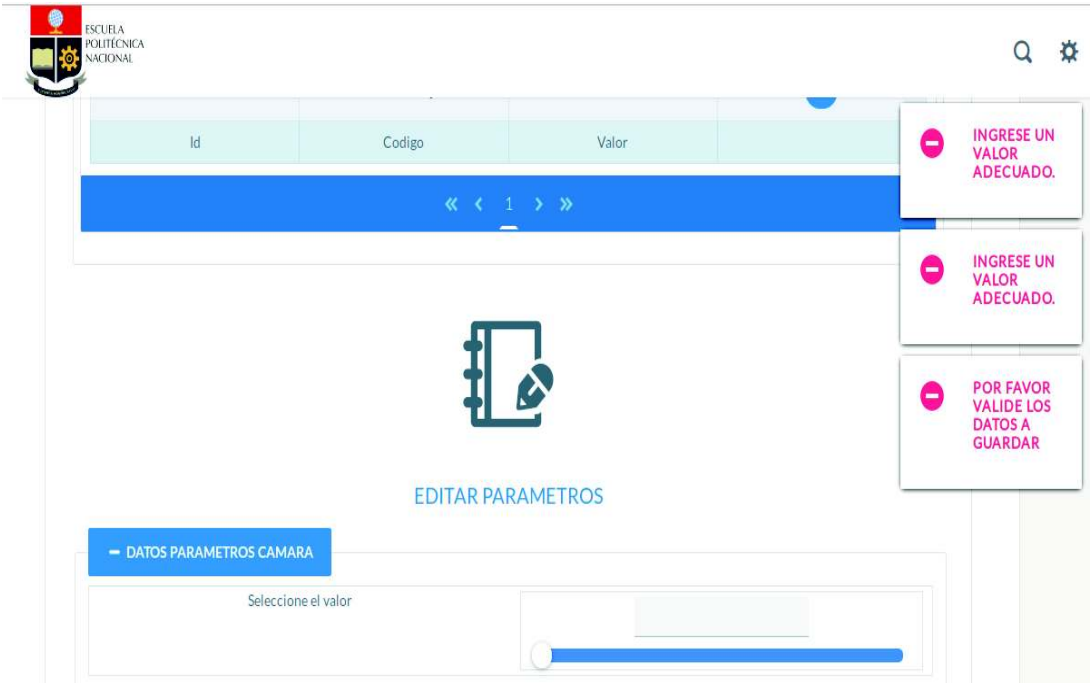


Figura 3.12 Captura de la no modificación de parámetros

A continuación en la Tabla 3.3 se puede observar los resultados obtenidos.

Tabla 3.3 Resultados generales funcionalidades de operador

Historia de usuario	criterio de aceptación	laptop	teléfono inteligente	observación
HUO001	Se crea una cámara	OK	OK	Para mejor visualización en el teléfono móvil la pantalla debe estar de forma horizontal
	No se crea una cámara	OK	OK	
	Se actualiza una cámara	OK	OK	
	No se actualiza una cámara	OK	OK	
	Se elimina una cámara	OK	OK	
HUO002	Se actualiza los parámetros	OK	OK	ninguna
	No se actualiza los parámetros	OK	OK	
HUO003	Se captura el stream en tiempo real	OK	OK	Verificación del funcionamiento de las cámaras
	No se captura el stream en tiempo real	OK	OK	
HUO004	Se procesa el stream	OK	OK	ninguna
	No se procesa el stream	OK	OK	
HUO005	Se guarda el conteo	OK	OK	ninguna
	No se guarda el conteo	OK	OK	

3.2.3 FUNCIONALIDADES DE USUARIO

Las pruebas de funcionalidad de usuario cubren 6 historias de usuario, cada una con 2 criterios de aceptación. A continuación se describen las historias más relevantes.

3.2.3.1 HUU001 - Visualizar cámaras

Esta historia de usuario consta de 2 criterios de aceptación los mismos que se describen a continuación.

Se visualiza la cámara. – En caso que se esté generando un conteo vehicular con el stream de esta cámara y se guarde en la base de datos, mediante una consulta

recursiva se podrá actualizar los datos de las cámaras, como se muestra en la Figura 3.13.

Figura 3.13 Visualización de cámaras en mapa en tiempo real

No se visualiza la cámara. – En caso de que las cámaras no esté funcionando debidamente o el módulo de recepción y procesamiento este parado, en este caso no se generará registros en la base de datos por lo cual al momento de realizar la consulta no se tendrá las coordenadas de la o las cámaras y no se mostrarán sobre el mapa.

3.2.3.2 HUU002 - Reportes de conteo

Esta historia de usuario consta de 2 criterios de aceptación los mismos que se describen a continuación.

Se generan reportes. – En caso de que la consulta realizada a la base de datos sea exitosa, e mostrará un gráfico estadístico como se observa en la figura 3.14.

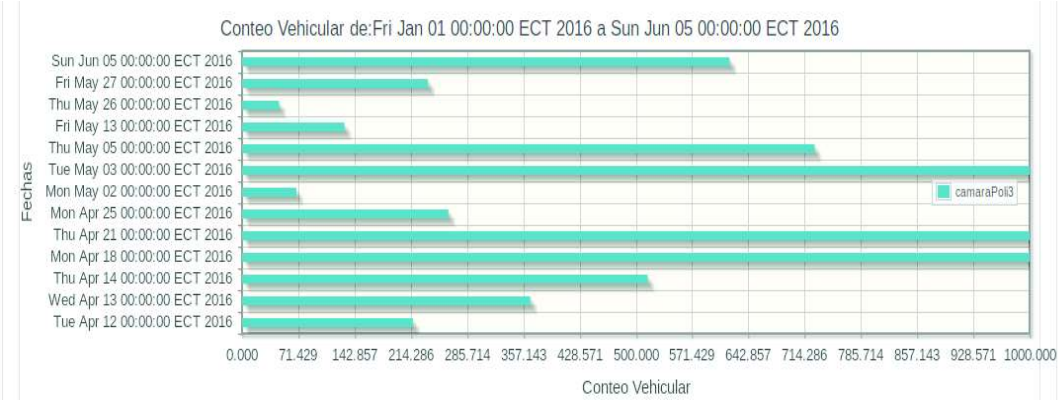


Figura 3.14 Generación de gráfico estadístico

No se generan reportes. – En caso de que caso de que la consulta realizada a la base de datos no sea exitosa o no se tengan datos referentes a los parámetros que ingrese el cliente.

3.2.3.3 HUU005 - Autenticación de usuarios

Esta historia de usuario consta de 2 criterios de aceptación que se describen a continuación.

Se obtiene el menú de usuario. – En caso de que el usuario se haya autenticado de manera correcta se desplegará los menús de usuario según se ya autenticado a la aplicación como se observa en la Figura 3.15, la Figura 3.16 y la Figura 3.17.

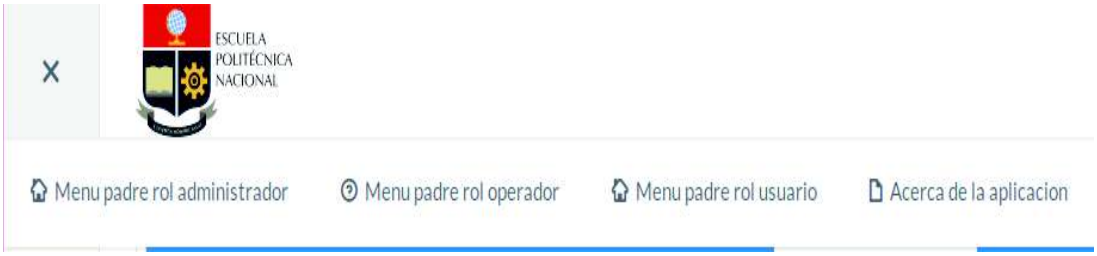


Figura 3.15 Menú de administrador

Figura 3.16 Menú de operador

Figura 3.17 Menú de usuario

No se obtiene el menú de usuario. – En caso de que el usuario no se haya autenticado de manera correcta como se observa en la Figura 3.18.

Figura 3.18 Login fallido

A continuación en la Tabla 3.4 se puede observar los resultados obtenidos.

Tabla 3.4 Tabla de resultados

Historia de usuario	criterio de aceptación	laptop	teléfono inteligente	observación
HUU001	Se visualiza la cámara	OK	OK	ninguna
	No se visualiza la cámara	OK	OK	
HUU002	Se generan reportes	OK	OK	ninguna
	No se generan reportes	OK	OK	
HUU003	Se crea una cuenta de usuario	OK	OK	En teléfonos inteligentes con pantalla de 4” se dificulta mucho la visualización
	No se crea una cuenta de usuario	OK	OK	
HUU004	Se actualiza una cuenta de usuario	OK	OK	ninguna
	No se actualiza una cuenta de usuario	OK	OK	
HUU005	Se obtiene el menú de usuario	OK	OK	Ninguna
	No se obtiene el menú de usuario	OK	OK	
HUU006	Se cargan los datos	OK	OK	Ninguna
	No se cargan los datos	OK	OK	

3.3 PRUEBAS DE INTEGRACIÓN

Se realizaron pruebas de integración entre los módulos para poder verificar su funcionamiento del prototipo completo, los módulos involucrados son los siguientes:

- (1) Módulo de captura y transmisión
- (2) Módulo de recepción y procesamiento
- (3) Módulo de almacenamiento
- (4) Módulo de reportes

En la Tabla 3.8 se detallan los casos de pruebas que se realizaron.

Tabla 3.5 Casos de pruebas de integración

Caso de Prueba	Módulos	Descripción de lo que se Probará	Prerrequisitos
CPI001	1- 2	Se probará la transmisión por streaming del video capturado por las cámaras y la posterior recepción del mismo para su procesamiento	Tener listo el escenario de pruebas
CPI002	2-3	Se probará el procesamiento del video capturado y él envió del conteo vehicular a la base de datos	Tener listo el escenario de pruebas, tener corriendo el motor de base de datos
CPI003	4-3	Se probará el funcionamiento de las operaciones CRUD que involucran estos dos sistemas	Autenticarse con un rol de usuario, tener corriendo el motor de base de datos
CPI004	1-2-3-4	Se probará los reportes de conteo vehicular que involucra los 4 módulos	Tener listo el escenario de pruebas, autenticarse con un rol de usuario, tener corriendo el motor de base de datos

3.3.1 CPI001 - TRANSMISIÓN DE STREAMING

Paso 1. – En este primer paso se comprueba el funcionamiento de las cámaras. Para su configuración y pruebas se empleó las aplicaciones web propias de las cámaras. En la Figura 3.19 y la Figura 2.20 correspondientes a las páginas de configuración pertenecientes a cada cámara.

Figura 3.19 Pantalla de configuración cámara entrada administración

Figura 3.20 Pantalla de configuración cámara entrada CEC

Paso 2. – El segundo paso se realiza en el módulo de recepción y procesamiento. Mediante el algoritmo desarrollado, se captura el stream producido por las cámaras en tiempo real, como se observa en la Figura 3.21 y Figura 3.22

Figura 3.21 Captura de video en tiempo real entrada administración

Figura 3.22 Captura de video en tiempo real entrada CEC

En la Tabla 3.6 se puede observar lo resultados de las pruebas de integración entre estos dos módulos.

Tabla 3.6 Resultado caso de prueba 1

CPI001				
Paso	**Descripción de pasos a seguir**	**Datos Entrada**	**Salida Esperada**	**¿OK?**
1	Paso 1: comprobar funcionamiento de la cámara	ninguno	Stream video	ok
2	Paso 2: verificar la recepción del stream de video	Stream video	Visualización de video	ok

3.3.2 CPI002 - CONTEO VEHICULAR

Esta prueba de integración se realiza entre el módulo de recepción y procesamiento y el módulo de almacenamiento, que consta de cuatro pasos que serán descritos a continuación.

Paso 1. – Se comprueba el conteo vehicular mediante el algoritmo desarrollado como se ve en la Figura 3.23.

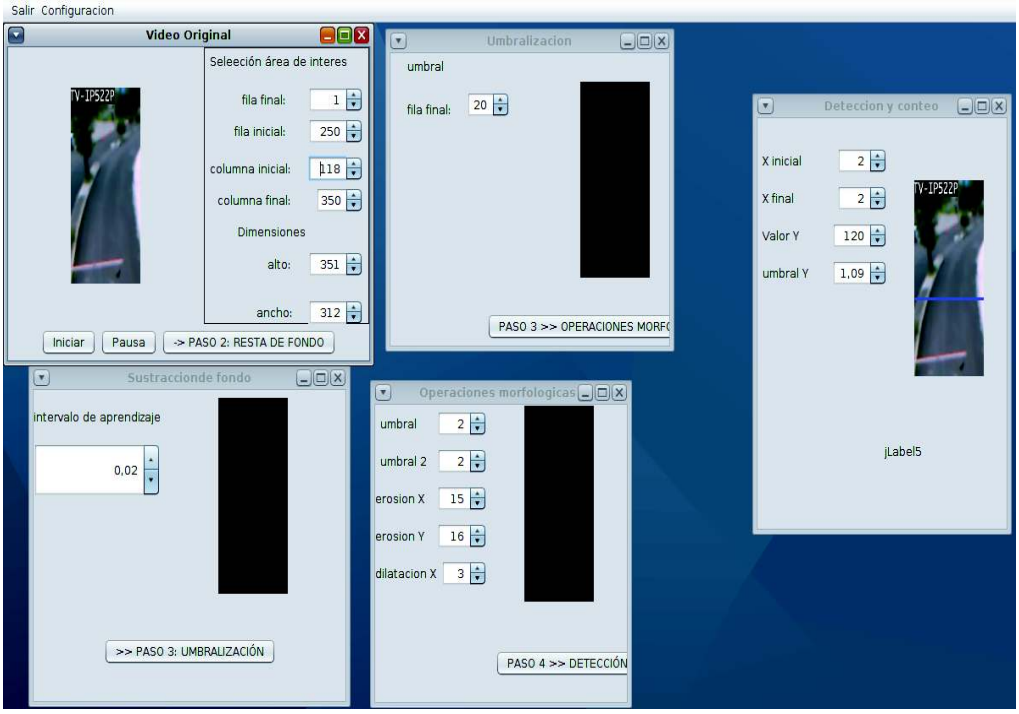


Figura 3.23 Generación de conteo vehicular de la entrada de administración

Paso 2. – Se verifica que los datos de conteo a almacenar en la BDD sean correctos, para ello se imprime por consola la cadena a almacenar. Ver Figura 3.24.

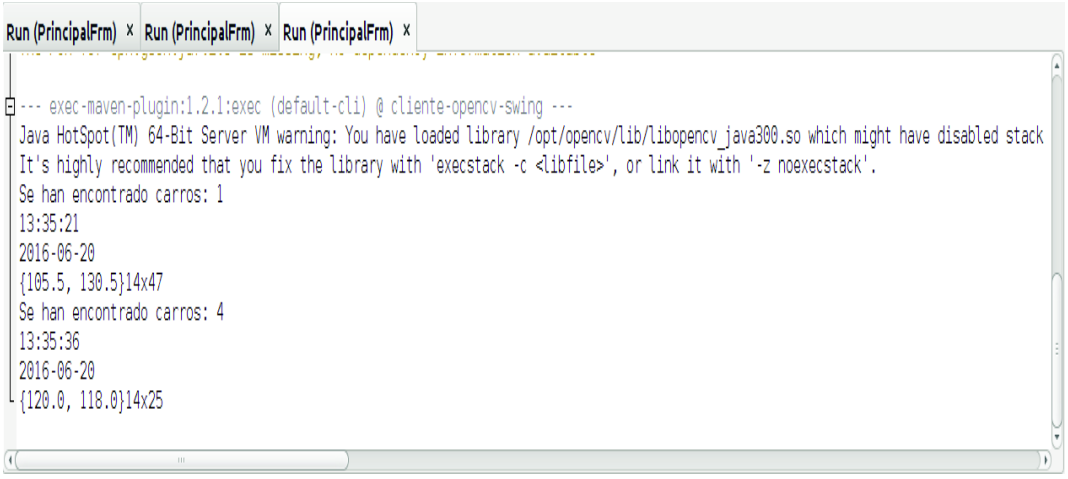


Figura 3.24 Datos enviados a guardar mostrado en consola

Paso 3. – mediante la utilización de la herramienta Poster se prueba los servicios Rest, como se puede observar en la Figura 3.25, en la cual se realiza la operación POST y la Figura 3.26 en la que se muestra la operación RESPONSE.

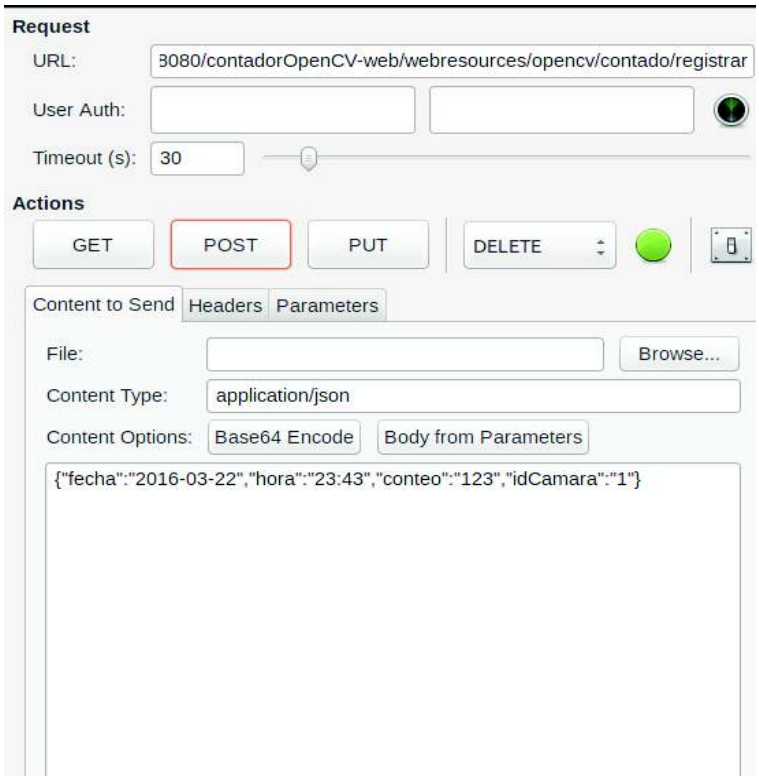


Figura 3.25 Operación POST

Figura 3.26 Operación RESPONSE

Paso 4. – Se realiza la verificación del almacenamiento del conteo vehicular en la base de datos. Para esto se han revisado los registros de la BDD, tal como se observa en la Figura 3.27.

Figura 3.27 Registros de conteo en la base de datos

En la Tabla 3.7 se puede observar lo resultados de las pruebas de integración entre estos módulos.

Tabla 3.7 Resultado caso de prueba 2

CPI002				
Paso	**Descripción de pasos a seguir**	**Datos Entrada**	**Salida Esperada**	**¿OK?**
1	Paso 1: Generar conteo vehicular	Stream de video proveniente de las cámaras	Conteo vehicular	ok
2	Paso 2: Enviar conteo vehicular	Conteo vehicular, hora de toma de muestra, fecha y código de identificación de la cámara	Cadena de caracteres del tipo String	ok
3	Paso 3. Recepción de la información enviada por Json	Cadena de caracteres del tipo String	Parámetros a ser guardados en la base de datos con sus respectivos tipos	ok
4	Paso 4: Guardado de conteo vehicular	Conteo vehicular, hora de toma de muestra, fecha y código de identificación de la cámara	Ninguna	ok

3.3.3 CPI003 - OPERACIONES CRUD

Esta prueba de integración se realiza entre el módulo de almacenamiento y el módulo de reportes, que consta de tres pasos que serán descritos a continuación.

Paso 1. – Se crea un usuario en la interfaz gráfica como se observa en la Figura 3.28, y se procede a verificar la creación en la BDD como se observa en la Figura 3.29.

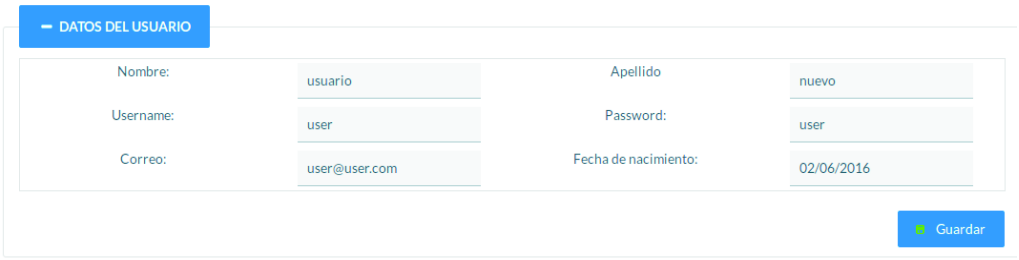


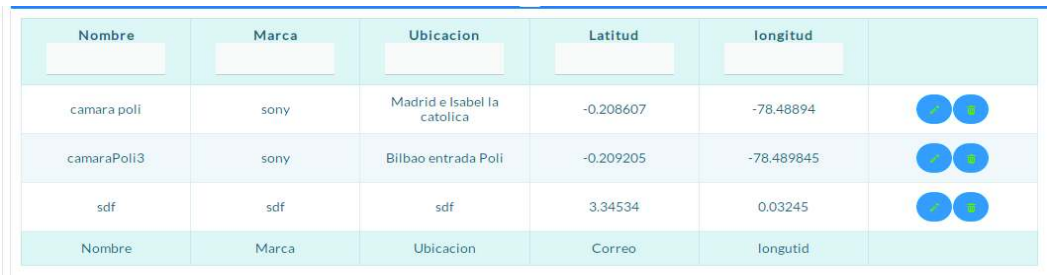
Figura 3.28 Creación de un usuario nuevo

Figura 3.29 Verificación de usuario nuevo en la base de datos

Paso 2. – Se realiza la actualización de uno de los datos seleccionados en la vista. Una vez realizado el cambio se actualiza la tabla de datos como se muestra en la Figura 3.30.

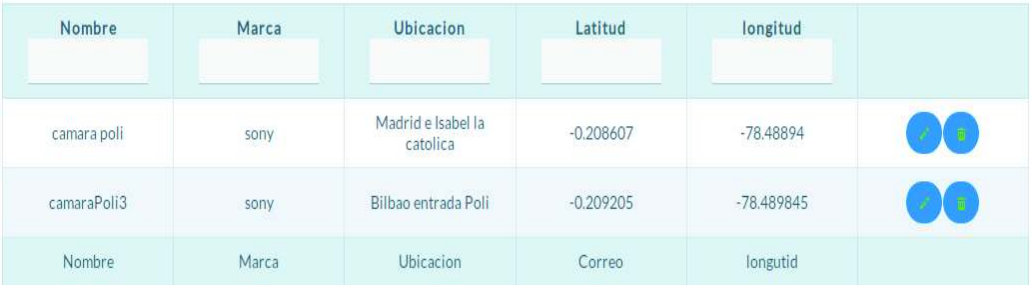
Figura 3.30 Tabla actualizada

Paso 3. – Se eliminar un registro de la tabla que se muestra en la figura 3.31. Después de realizada la operación solo quedan dos registros como se observa en la figura 3.32.



Nombre	Marca	Ubicacion	Latitud	longitud	
camara poli	sony	Madrid e Isabel la catolica	-0.208607	-78.48894	
camaraPoli3	sony	Bilbao entrada Poli	-0.209205	-78.489845	
sdf	sdf	sdf	3.34534	0.03245	
Nombre	Marca	Ubicacion	Correo	longutid	

Figura 3.31 Tabla sin eliminar un parámetro



Nombre	Marca	Ubicacion	Latitud	longitud	
camara poli	sony	Madrid e Isabel la catolica	-0.208607	-78.48894	
camaraPoli3	sony	Bilbao entrada Poli	-0.209205	-78.489845	
Nombre	Marca	Ubicacion	Correo	longutid	

Figura 3.32 Tabla con un parámetro eliminado

En la Tabla 3.811 se puede observar lo resultados de las pruebas de integración entre estos módulos.

Tabla 3.8 Resultado caso de prueba 3

CPI003				
Paso	**Descripción de pasos a seguir**	**Datos Entrada**	**Salida Esperada**	**¿OK?**
1	Paso 2:realizar operación de crear	Datos solicitados por la aplicación	Inserción en la base de datos	ok
2	Paso 3. Realizar operación de actualizar	Datos elegidos por el usuario	Actualización de base de datos	ok
3	Paso 4: realizar operación de eliminar	Datos elegidos por el usuario	Actualización de base de datos	ok

3.3.4 CPI004 - CONSULTA REPORTES

Esta prueba de integración se realiza entre el módulo de almacenamiento y el módulo de reportes.

Se prueba el funcionamiento de los reportes de conteo sobre el mapa, como se muestra en la Figura 3.33 y los reportes sobre un gráfico de barras como se observa en la en la Figura 3.34.

Figura 3.33 Reporte en tiempo real de conteo vehicular

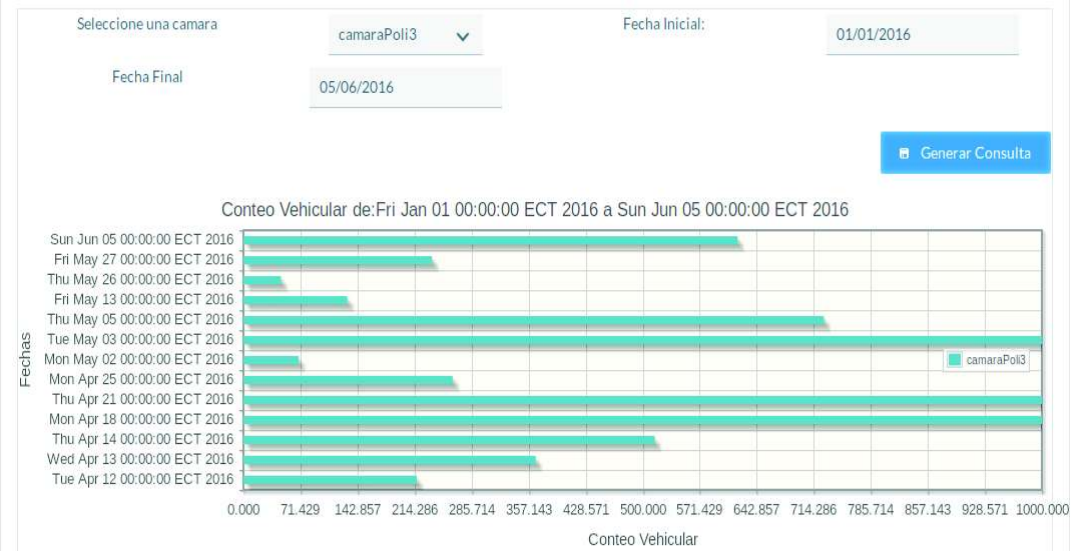


Figura 3.34 Reporte en tiempo histórico de conteo vehicular

En la Tabla 3.12 se puede observar lo resultados de las pruebas de integración entre estos módulos.

Tabla 3.9 Resultado caso de prueba 4

CPI004				
Paso	**Descripción de pasos a seguir**	**Datos Entrada**	**Salida Esperada**	**¿OK?**
1	Paso 1:realizar la consulta de información	Datos solicitados por la aplicación	Resultado de la consulta	ok

3.4 ANALISIS DE RESULTADOS DEL ALGORITMO DE CONTEO VEHICULAR EN TIEMPO REAL

A continuación se realiza un análisis de los resultados obtenidos.

3.4.1 RESULTADOS DURANTE EL DESARROLLO

El desarrollo del algoritmo de conteo se realizó en base a un video de prueba que muestra un tramo de una autopista donde el tráfico es fluido, el video se encuentran adjuntos en el Anexo D.1. Con este video se desarrolló una primera versión del algoritmo de conteo, obteniéndose los resultados mostrados en la Tabla 3.10.

Tabla 3.10 Resultados videos de prueba para desarrollo

Video de prueba	**Conteo manual**	**Conteo con el algoritmo**	**Porcentaje de error**
video1	41	37	9.74%

El porcentaje de error fue calculado con la siguiente formula:

- %error: [(|Conteo manual-Conteo automático|) / Conteo manual] x 100

Los valores mostrados en la tabla 3.13. Fueron los mejores resultados obtenidos después de afinar y mejorar el algoritmo para este caso en particular (video pregrabado).

3.4.1.1 Análisis de errores

El error indicado en la sección anterior se debe principalmente a la perspectiva con la que se colocaron las cámaras, tal como se observa en la Figura 3.50.

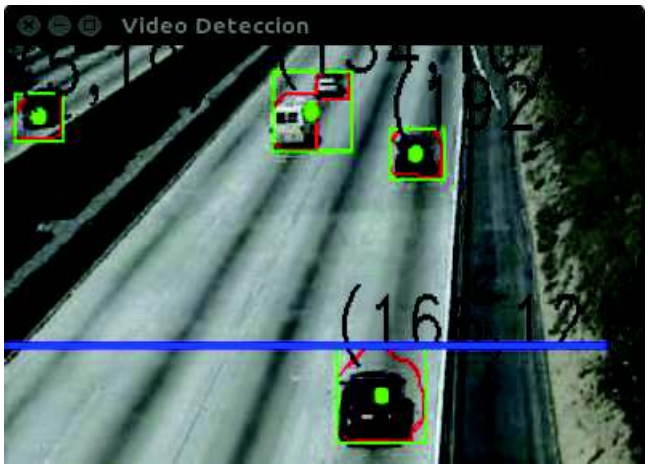


Figura 3.35 Escena del video1

A continuación, se lista los problemas y sus respectivas soluciones que se obtuvieron aplicando el algoritmo sobre estos videos:

Problema. – Vehículos muy juntos se detectan como uno solo.

Problema. – Vehículos muy grandes se detectan por partes.

Solución. – La solución contemplada para los dos primeros problemas fue aumentar la erosión del objeto de interés, lo suficiente como para lograr la independencia de cada uno de los objetos, en la medida de lo posible sin afectar la detección.

Problema. – Debido al número de Frames por segundo existen casos en los que un vehículo es correctamente detectado, pero no es contado debido a que los pixeles de su centro nunca coinciden con los pixeles establecidos en la línea de conteo.

Solución. – La solución para el tercer problema fue utilizar una zona de conteo en vez de la línea para así evitar el problema del no conteo del centro del objeto de interés. Al realizar esta acción se tiene una contradicción, ya que si la zona de conteo es demasiado grande, los cetros de los objetos de interés podrían contarse más de una vez.

3.4.2 RESULTADOS EN EL ESCENARIO EN TIEMPO REAL

Como se mencionó anteriormente, las cámaras fueron ubicadas en los ingresos de la EPN: ingreso de Administración ubicado en la calle Ladrón de Guevara y en el ingreso al subsuelo del Edificio de Aulas y Relación con el Medio Externo ubicado en la calle Toledo. La entrada a la EPN se realiza de dos formas: mediante el TAG para entrar automáticamente y mediante identificación manual con el guardia. Los resultados del algoritmo de conteo vehicular en tiempo real se observa en la Tabla 3.14.

Tabla 3.11 Resultados conteo vehicular en tiempo real

Ubicación de la cámara	Conteo manual	Conteo con el algoritmo	Porcentaje de error
Entrada administración	22	25	12%
Entrada cec	7	7	0%

El porcentaje de error fue calculado con la siguiente formula:

- %error: [(|Conteo manual-Conteo automático|)/Conteo manual] x 100

Los valores mostrados fueron tomados en el horario de 8:00 a 8:30, se decidió tomar este intervalo de prueba debido a que existe mayor ingreso de vehículos a la institución.

3.4.2.1 Análisis de errores

A continuación, se lista los problemas y sus respectivas soluciones que se obtuvieron aplicando el algoritmo sobre estos videos:

Se tuvieron errores en el conteo vehicular por que se presentaron los siguientes inconvenientes:

> **Problema. –** El primer caso es cuando los automóviles entran con el TAG, lo hacen a una velocidad considerable, esto en conjunto con los frame por segundos definidos en la cámara, genera que los pixeles del centro del objeto de interés nunca coincidían con la línea de conteo.

Solución. – La solución para el problema fue utilizar una zona de conteo en vez de la línea para así evitar el problema del no conteo del centro del objeto de interés. Al realizar esta acción se tiene la misma contradicción que en la solución expuesta al problema similar en la etapa de desarrollo.

Problema. – El segundo caso es cuando los vehículos no disponen del TAG, tienen que detenerse e identificarse para ingresar.

En este caso los vehículos arrancan lentamente; consecuentemente el centro detectado coincidía varias veces con la línea de conteo y el contador aumenta varias veces.

Solución. – La solución a este problema fue muy dificultosa debido en gran parte a que para solucionar el problema anterior se hace uso de la zona de conteo. Al implementar esto se aumenta el margen de error del presente problema.

La solución propuesta fue el paralelizar la aplicación con el uso de threads o hilos y Executors. El proceso es el siguiente:

- Existe un hilo principal que ejecuta todo el algoritmo menos la fase de detección.
- En la fase de detección se implementa la interface **java.util.concurrent.ExecutorService** esta incorpora un mecanismo de ejecuciones asíncronas para ejecutar tareas en paralelo con el hilo principal, con la definición de un pool de hilos.
- Se define un pool de 10 hilos, los cuales serán invocados al momento de que el contador sume en uno. Uno de estos hilos será el encargado de realizar la acción del contador sume en uno, al hacer esto se invoca el método Sleep sobre el mismo durante un periodo de tiempo. El resto de hilos se encarga de dar continuidad al video para que esta acción sea transparente.
- Después de trascurrido el tiempo definido para que el automóvil pase por la zona de conteo, el hilo devuelve el valor del contador. Inmediatamente

se termina la invocación del pool y se espera por el siguiente conteo esto permite solventar el presente problema e invalida la contradicción que existía en el primer problema.

Problema. – Otro de los errores es causado por los transeúntes que cruzan por media calle y son detectados como vehículos; por lo tanto, si pasan por la línea de conteo son contados como si se tratase de un automóvil.

Solución. – La solución propuesta para el problema, fue ubicar la zona de conteo en la parte más baja de la imagen de cada cámara, ya que se determinó que en estas zonas casi nunca hay alguna interferencia a o paso de personas.

Problema. – El último problema es el error generado debido a los obstáculos, y las sombras generadas por los propios vehículos. Durante el día las sombras son grandes por ejemplo, en las mañanas las sombras de los transeúntes se pueden observar sobre la calzada. Dichas sombras, al estar en movimiento también son detectadas y contadas al pasar por la línea de conteo.

Solución. –La solución propuesta para el tercer problema, fue aumentar la erosión del objeto de interés lo suficiente como para que el algoritmo no detecte el contorno de una persona, ni las sombras producidas por estos.

CAPÍTULO 4: CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presentan las conclusiones y las recomendaciones entorno al trabajo realizado, basado en los conocimientos adquiridos a lo largo del diseño e implementación del mismo.

4.1 CONCLUSIONES

El presente proyecto presenta un prototipo de un sistema web para conteo de vehículos en tiempo real, a través del procesamiento de imágenes en tiempo real mediante la librería de visión artificial de código abierto OPENCV, el lenguaje de programación JAVA y el framework Java Server Faces.

El presente trabajo pretende ser la base para un sistema que permita determinar el flujo vehicular en una calle o avenida, mediante el uso de las cámaras instaladas en ciertas ciudades. Esta solución permitiría reducir los tiempos de viaje, a un costo de implementación bajo en comparación con sistemas de monitoreo de tráfico propietarios.

El diseño modular del prototipo facilitó la separación de funciones, además deja la posibilidad de adjuntar otros módulos con nuevas funcionalidades con un ajuste mínimo entre estos o modificaciones sobre los módulos existentes si se desea brindar nuevos servicios.

Para realizar el conteo vehicular se usó una línea de conteo. Esto se logró utilizando las coordenadas en pixeles del centro de los objetos en movimiento, de tal forma que si los pixeles de la línea de conteo coinciden con los pixeles del centro del objeto el contador sube en 1. La efectividad de este conteo depende enteramente de los Frames por segundo que tenga el video.

El algoritmo desarrollado para el conteo vehicular depende de muchos factores como la posición de la cámara, factores medio ambientales como la iluminación, el clima, la ubicación, las sombras y la velocidad del viento.

El sistema implementado es del tipo web, para así motivar a los usuarios a utilizar el sistema únicamente a través de un navegador, garantizando la compatibilidad del mismo con cualquier dispositivo móvil o de escritorio conectado al Internet o red interna en el cual este corriendo el sistema.

El algoritmo propuesto para el conteo vehicular en su fase de desarrollo presentó una efectividad del 91% referente al número de vehículos contados exitosamente. Los errores correspondientes a falsos positivos y negativos son debido a problemas de colocación y ángulo de la cámara. Además, el algoritmo detecta como un solo cuerpo a los vehículos que transitan muy juntos, y esto ocasiona un falso negativo.

En las pruebas efectuadas en tiempo real se verificó que la efectividad del algoritmo de conteo vehicular en los accesos de la Escuela Politécnica Nacional depende exclusivamente de los obstáculos (transeúntes, sombras en movimiento) en el área de conteo, ya que estos generan falsos positivos y negativos en el conteo. La solución hallada para corregir estos errores en las entradas vehiculares, involucró la implementación de paralelismo de tareas a través del uso de hilos asíncronos. Esto fue posible gracias al uso de la interface **java.util.concurrent.ExecutorService.** Gracias a esto la efectividad del conteo vehicular en el acceso de la calle Ladrón de Guevara es del 88% y el del acceso de la Av. Toledo es del 100% con respecto al conteo exitoso de vehículos, estos valores son variables ya que dependen de algunos factores.

El uso del framework Java Server Faces contribuyo sobremanera en el desarrollo del módulo de reportes ya que este exhibe un marco de trabajo que simplifica el desarrollo de interfaces para aplicaciones Java EE. Además, ligado a la librería de componentes PrimeFaces proporciona un conjunto muy diverso de componentes y plantillas, con los cuales se desarrolló y personalizo la vista del sistema web.

El desarrollo del presente trabajo fue basado en la visión artificial, y el aporte que se extiende del mismo es su utilización por parte de la Dirección de Gestión de la Información y Procesos (DGIP), la misma que necesitaba tener un control del número de vehículos que ingresan a los parqueaderos de la Escuela Politécnica Nacional. El prototipo implementado consiste de dos cámaras, cada una ubicada

en los accesos vehiculares de: la calle Ladrón de Guevara, y en la Av. Toledo. Para consultar la información del conteo de vehículos se utiliza la aplicación web desarrollada como parte del prototipo, y con la cual el personal de seguridad podrá implementar métodos y técnicas que ayuden a la mejor distribución de los vehículos por los diferentes accesos de la institución.

4.2 RECOMENDACIONES

Al momento de elegir la ubicación de la cámara para optimizar el conteo esta debería ser lo más perpendicular posible hacia la calle ya que así se eliminaría completamente el efecto de sombra que proviene de los mismo vehículos. Así como la de los obstáculos, que se tiene cuando está se encuentra en una posición de inclinación hacia la calle. Además se debe asegurar la posición estática de las mismas, ya que el más mínimo movimiento ocasionaría algún mal funcionamiento del algoritmo de conteo.

Antes de poner en funcionamiento el módulo de recepción y procesamiento se debe verificar el funcionamiento físico de las cámaras, además de verificar si el servicio de streaming de las misma está funcionando ya sea por HTTP o RTSP y por último, pero no menos importante, verificar que el servidor Glassfish este corriendo para que se puedan guardar los datos de conteo en la base de datos.

Para futuros proyectos o mejoras basadas en este prototipo, se podrá agregar a la detección de objetos el seguimiento de ruta y el reconocimiento de objetos para así aumentar los parámetros como la velocidad del automóvil, dirección, tamaño, etc. Con lo cual se podría brindar más precisión al estimado de flujo vehicular en determinada calle o avenida.

REFERENCIAS BIBLIOGRÁFICAS

[1] J. J. Gibson," The Ecological Approach To Visual Perception", Edición: Revised. Psychology Press, 1986.

[2] D. Marr, "Vision: A Computational Investigation Into the Human Representation and Processing of Visual Information". London, 2010.

[3] Anónimo, “Visión artificial e interacción sin mandos”. [En línea]. Disponible en: http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/3D/VisionArtificial/index.html. [Accedido: 17-mar-2016].

[4] J. Vélez, A. Moreno “Libro sobre Visión Artificial - Visión por Computador”. [En línea]. Disponible en: http://www.visionporcomputador.es/libroVision/libro.html. [Accedido: 02-mar-2016].

[5] Anónimo, “Tweak And Trick - Technology Blog”. [En línea]. Disponible en: http://www.tweakandtrick.com/. [Accedido: 04-feb-2016].

[6] G. Bradski y A. Kaehler, "Learning OpenCV: Computer vision with the OpenCV library". O’Reilly Media, Inc., 2008.

[7] Itseez, “OpenCV | OpenCV”, OPENCV - Open Source Computer Vision. [En línea]. Disponible en: http://opencv.org/. [Accedido: 19-feb-2016].

[8] L. De Russis y A. Sacco, “Welcome to OpenCV Java Tutorials documentation! — OpenCV Java Tutorials 1.0alpha documentation”. [En línea]. Disponible en: http://opencv-java-tutorials.readthedocs.org/en/latest/. [Accedido: 18-feb-2016].

[9] A. J. M. Sierra, "Programador Certificado JAVA 2. Curso práctico". 3a Edición. RA-MA S.A. Editorial y Publicaciones, 2010.

[10] Oracle Corporation, “- The Java EE 6 Tutorial”. [En línea]. Disponible en: http://docs.oracle.com/javaee/6/tutorial/doc/. [Accedido: 07-mar-2016].

[11] Anónimo, “Introducción a JavaServer Faces”. [En línea]. Disponible en: http://www.jtech.ua.es/j2ee/publico/jsf-2012-13/sesion01-apuntes.html. [Accedido: 16-mar-2016].

[12] P. Alcantarrilla, “Sistema de monitorización y control de tráfico en carretera”, Ingenieria, Universidad de Alcalá, madrid, 2006.

[13] F. Calderon y Urrego German, “Conteo automático de vehículos”, Ingenieria, Pontificia Universidad Javeriana, Bogota, 2008.

[14] P. Valdés y M. Alejandro, “Algoritmo para conteo vehicular en tiempo real con base en franjas de interés”, 2010.

[15] H. Takeuchi y I. Nonaka, “The new new product development game”, Harvard business review, vol. 64, nº 1, pp. 137–146, 1986.

[16] I. Torras, “Introducción — Scrum 1 documentation”. [En línea]. Disponible en: http://metodologiascrum.readthedocs.org/en/latest/Scrum.html. [Accedido: 26-abr-2016].

[17] A. Peralta, “Metodología Scrum”, Universidad en Uruguay. [En línea]. Disponible en: http://fi.ort.edu.uy/innovaportal/file/2021/1/scrum.pdf. [Accedido: 26-abr-2016].

[18] pmoinformatica.com, “Plantillas Scrum: historias de usuario y criterios de aceptación - La Oficina de Proyectos de Informática”. [En línea]. Disponible en: http://www.pmoinformatica.com/2012/10/plantillas-scrum-historias-de-usuario.html. [Accedido: 26-abr-2016].

[19] E. Montero, “Diseño e implementación de un sistema web para el monitoreo del estado de un motor”, Tesis, Quito, 2016.

[20] J. Gargaz, “Como estimamos proyectos Scrum o en general ágiles - Javier Garzás”. [En línea]. Disponible en: http://www.javiergarzas.com/2014/01/estimacion-agil-scrum.html. [Accedido: 14-jul-2016].

[21] pmoinformatica.com, “Plantillas Scrum: Pila de producto (Product Backlog) - La Oficina de Proyectos de Informática”. [En línea]. Disponible en: http://www.pmoinformatica.com/2013/11/plantillas-scrum-pila-producto-product.html. [Accedido: 26-abr-2016].

[22] M. Gibert y A. Peña, “Ingeniería del Software en Entornos de Software Libre”. Fundació per a la Universitat Oberta de Catalunya. Primera edición. Marzo 2005

[23] TRENDnet, “TRENDnet | Products | TV-IP522P | Cámara de Internet PoE ProView Megapixel”. [En línea]. Disponible en: https://www.trendnet.com/langsp/products/proddetail?prod=170_TV-IP522P. [Accedido: 16-jul-2016].

[24] DAHUA TECHNOLOGY, “DH-IPC-HFW2300RN-Z&VF.pdf”, DAHUA, 16-feb-2016. [En línea]. Disponible en: http://www.dahuasecurity.com/en/us/download/DH-IPC-HFW2300RN-Z&VF.pdf. [Accedido: 16-jul-2016].

[25] D. L. Baggio, "OpenCV 3.0 computer vision with Java: create multiplatform computer vision desktop and web applications using the combination of OpenCV and Java". Packt Publishing Limited. 2015.

[26] The Apache Software Foundation, “Maven – Welcome to Apache Maven”. [En línea]. Disponible en: https://maven.apache.org/. [Accedido: 04-ene-2016].

[27] The Apache Software Foundation, “Archiva – The Build Artifact Repository Manager”. [En línea]. Disponible en: https://archiva.apache.org/index.cgi. [Accedido: 13-may-2016].

[28] The PostgreSQL Global Development Group, “PostgreSQL: The world’s most advanced open source database”. [En línea]. Disponible en: https://www.postgresql.org/. [Accedido: 26-may-2016].

[29] SQL Power Group, “Data Modeling & Profiling Tool: SQL Power Architect | SQL Power Software”. [En línea]. Disponible en: http://www.sqlpower.ca/page/architect#6186. [Accedido: 26-may-2016].

[30] Anónimo, “pgAdmin: PostgreSQL administration and management tools”. [En línea]. Disponible en: https://www.pgadmin.org/. [Accedido: 26-may-2016].

[31] Oracle Corporation, “JavaServer Faces Technology”. [En línea]. Disponible en: http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html. [Accedido: 26-may-2016].

[32] PrimeTek, “PrimeFaces”. [En línea]. Disponible en: http://primefaces.org/. [Accedido: 26-may-2016].

[33] Oracle Corporation, “GlassFish Server”. [En línea]. Disponible en: https://glassfish.java.net/. [Accedido: 26-may-2016].

[34] C. Caules, “Modulos de Java (III) Enterprise Archive (EAR)”, Arquitectura Java, [En línea]. Disponible en: http://www.arquitecturajava.com/ear/. [Accedido: 06-may-2016]

[35] C. Álvarez, “Patrones de Diseño (Active Record vs DAO)”, Genbeta Dev, 31-jul-2014. [En línea]. Disponible en: http://www.genbetadev.com/java-j2ee/patrones-de-diseno-active-record-vs-dao. [Accedido: 11-may-2016].

ANEXOS

ANEXO A: MANUALES DE USUARIO.

ANEXO B: DOCUMENTACIÓN SCRUM.

ANEXO C: CODIGO FUENTE DEL PROTOTIPO.

ANEXO D: VIDEOS DE PRUEBA.

Nota: Todos los ANEXOS se encuentran en el CD adjunto.