Identificación de pasajeros para el cobro de tarifas en transporte terrestre intermunicipal en Colombia

Jair Hernando Vidal Zúñiga

Nota de Aceptación

Certificamos que el presente Trabajo de Grado
Satisface, en alcances y calidad, todos los requisitos
Que demanda un Trabajo de Grado de Maestría.

Gloria Inés Alvarez V.
Director

Hernan Vargas
Jurado

María Constanza Pabon
Jurado

Aprobado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana Cali, para optar el título de Magister en Ingeniería de Software.

HERNÁN CAMILO ROCHA NIÑO Ph. D.
Decano Facultad de Ingeniería y Ciencias

JUAN CARLOS MARTÍNEZ ARIAS
Director Posgrados de Ingeniería y Ciencias

Santiago de Cali, 09/12/2020

Maestría en Ingeniería
Facultad de Ingeniería y Ciencias

Pontificia Universidad JAVERIANA Cali

Acta de Correcciones al Documento de Trabajo de Grado

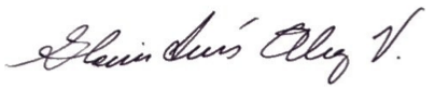
Santiago de Cali, 04 de diciembre de 2020

Autor**: Jair Hernando Vidal Zúñiga**

Título del Trabajo de Grado**: "Identificación de pasajeros para el cobro de tarifas en transporte terrestre intermunicipal en Colombia"**

Director**: Gloria Inés Álvarez**

Como indica el artículo 2.13 de las Directrices para Trabajo de Grado de Maestría, he verificado que el estudiante indicado arriba ha implementado todas las correcciones que los Jurados del Proyecto de Trabajo de Grado definieron que se efectuaran, como consta en el Acta de Evaluación correspondiente.



Firma del director del Trabajo de Grado

IDENTIFICACIÓN DE PASAJEROS PARA EL COBRO DE TARIFAS EN TRANSPORTE TERRESTRE INTERMUNICIPAL EN COLOMBIA

JAIR HERNANDO VIDAL ZUÑIGA
Código 8943741

Proyecto de trabajo de grado para optar al título de
Magister en Ingeniería de Software

Directora
GLORIA INÉS ÁLVAREZ V

FACULTAD DE INGENIERÍA
MAESTRÍA EN INGENIERÍA DE SOFTWARE
SANTIAGO DE CALI
2020

RESUMEN

Este trabajo de grado permite abordar una problemática que sucede a menudo en el transporte público intermunicipal y tiene que ver con el cobro de tarifas excesivas cuando un usuario toma el vehículo en un punto intermedio del recorrido y se baja antes que éste llegue a su destino final. En este contexto es necesario identificar al pasajero que se baja y saber dónde inició su recorrido previamente; esto, permite establecer la cantidad de kilómetros recorridos con el fin de calcular el costo que debe cancelar el usuario al llegar a su destino y notificar el valor a pagar por el servicio prestado.

El objetivo de este trabajo de grado fue desarrollar un prototipo funcional que permita identificar al pasajero a través de técnicas de biometría, calculando el costo de la tarifa y notificando el valor que debe cancelar el usuario al finalizar su viaje.

Se aplicaron todas las etapas de desarrollo de software y fue implementado utilizando el lenguaje de programación Python versión 3.7.2, utilizando conceptos de programación orientada a objetos y patrones de diseño; adicionalmente, contó con un repositorio de código Git.

El prototipo cuenta con 3 componentes de software, que vale la pena destacar, los cuales son: un componente que controla la captura de imágenes, un componente que identifica al pasajero; un componente que permite calcular la tarifa con base a la distancia recorrida por el usuario y el componente que permite notificar al pasajero acerca del valor a cancelar.

Con una base de datos de imágenes diversa, se realizan diferentes pruebas y con los resultados se obtuvo que el algoritmo de patrones binarios locales utilizado en el reconocimiento facial disminuye su efectividad cerca del 20% debido a cambios en la captura de la imagen o la posición del rostro.

ABSTRACT

This degree work allows us to tackle a problem that often occurs in inter-municipal public transport and has to do with the charging of excessive rates when a user takes the vehicle at an intermediate point of the route and gets off before it reaches its final destination. In this context, it is necessary to identify the passenger who gets off and know where they had previously gotten on, this allows establishing the number of kilometers traveled in order to calculate the cost that the user must pay when arriving at their destination and notify the amount to pay for the service provided.

The objective of this degree work was to develop a functional prototype that allows the passenger to be identified through biometric techniques, calculating the cost of the fare and notifying the amount that the user must pay at the end of their trip.

All the software development stages were applied, and it was implemented using the Python programming language version 3.7.2, using object-oriented programming concepts and design patterns, additionally it had a Git code repository.

The prototype has 3 software components, which are worth highlighting, which are: a component that controls the capture of images, a component that identifies the passenger; a component that allows the fare to be calculated based on the distance traveled by the user and the component that allows notifying the passenger about the amount to be canceled.

With a diverse image database, different tests are carried out and with the results it was obtained that the local binary pattern algorithm used in facial recognition decreases its effectiveness by about 20% due to changes in the image capture or position of the face.

TABLA DE CONTENIDO

LISTA DE FIGURAS

Pontificia Universidad JAVERIANA Cali

Res. 2333 del 2012

LISTA DE TABLAS

GLOSARIO

En esta sección se enuncian algunos términos que son usados en el contexto del transporte público y que son importantes para mejorar la comprensión del proyecto.

Acompañante: Persona que viaja en el mismo automotor y realiza el recaudo de los pasajes.

Compañía de transporte público: Empresa que presta el servicio de transporte terrestre intermunicipal.

Conductor: Persona que conduce bus de transporte terrestre intermunicipal.

Confianza: Medida de distancia que indica, por medio de la fórmula de distancia entre dos puntos, que tan parecidos son dos rostros.

Multitarifa: Valores en dinero asociados a cada uno de los tramos con que cuenta una ruta.

Pasaje o tarifa: Valor que debe ser cancelado por el pasajero y que corresponde a costo por el servicio de transporte.

Pasajero o usuario: Persona que aborda un bus en una terminal terrestre con un destino definido.

Ruta de transporte terrestre intermunicipal: Conexión terrestre entre dos o más puntos geográficos normalmente ciudades principales o intermedias.

Tramo o tramo de ruta: Se define como la distancia recorrida por un pasajero dentro de una ruta de transporte terrestre intermunicipal. Ejemplo: la ruta Cali – Bogotá, cuenta con varios tramos entre ellos: Cali – Buga, Cali – Armenia, Armenia – Ibagué entre otros.

INTRODUCCIÓN

El transporte público en las grandes urbes del mundo es un servicio que contribuye al desplazamiento de las personas y al medio ambiente, disminuyendo las emisiones de gases, mejorando la salud del planeta.

En Colombia el sector del transporte público intenta prestar el mejor servicio a sus usuarios, pero en ocasiones la evasión de pasajes por parte de algunos usuarios aumenta la falta de recursos y hace que no se pueda cumplir con la demanda de pasajeros.

Por otro lado, aunque el transporte público intermunicipal en Colombia tiene libertad de tarifas[1], en ocasiones no se reportan los cobros que se hacen en rutas o tramos cortos, pues es difícil identificar si una persona inicia su viaje en un punto y desciende en otro. Sumado a esto, calcular el valor exacto de la tarifa es un poco complicado.

En los últimos años el uso de tecnologías de identificación biométrica, han permitido reducir sustancialmente la suplantación de personas en contextos como comicios electorales, la seguridad y la vigilancia, ésta tecnología combinada con otras como tarjetas inteligentes hacen casi imposible evadir los controles que realizan las instituciones del gobierno o la fuerza pública; la aplicación de técnicas de biometría en el transporte público intermunicipal en la actualidad es muy reducida por que se pueden presentar múltiples factores que podrían alterar las información capturada y producir respuestas equivocadas; sin embargo, es un mercado que tiene mucho potencial, debido al creciente uso de unidades GPS, contadores y cámaras de video en vehículos de transporte público, se hace necesario desarrollar dispositivos que combinen la biometría con los elementos antes mencionados, con el fin de permitir el recaudo de pasajes de una manera segura y confiable.

El implementar este software como servicio en un modelo de negocio, representa una revolución en la forma de ofrecer servicios; y, por lo tanto, como los clientes

acceden a éstos, de la misma manera se puede implementar este modelo en el transporte público apoyado en el uso de la biometría, de modo que los usuarios puedan acceder a tarifas justas por kilómetro o metro recorrido con el propósito de mejorar el cobro de tarifas en el transporte público intermunicipal. En este trabajo de grado se desarrolló un prototipo que permite calcular la tarifa exacta que debe asumir el usuario por el servicio prestado tomando como referencia la cantidad de kilómetros recorridos desde el inicio y hasta el fin del viaje.

Con este trabajo de grado se busca mejorar el servicio, optimizando los tiempos de desplazamiento, establecer un cobro exacto y aumentar la productividad de las empresas de transporte intermunicipal.

1. DEFINICIÓN DEL PROBLEMA

1.1. Planteamiento del problema

En Colombia, el transporte terrestre intermunicipal cuenta con muchos problemas: la falta de vías en óptimas condiciones, los costos elevados del combustible, los insumos para su operación, sumado a esto, el incremento en las tarifas, entre otros aspectos que afectan directamente el bolsillo de los usuarios.

Un viaje en transporte terrestre intermunicipal inicia con la compra de tiquetes en la terminal de transporte de la ciudad origen; el pasajero aborda el automotor y finalizará en la terminal de destino ubicada en una ciudad que normalmente es capital de otro departamento, para citar un ejemplo: un viaje de Cali a Bogotá; implica un trayecto de más de 8 horas, durante el cual existen múltiples destinos cortos, es decir, ciudades intermedias o pueblos y por lo general, la gran mayoría de los pasajeros no viajan la ruta completa (ruta Cali - Bogotá) cuyo desplazamiento concluye en ciudades intermedias (Buga, Armenia, Ibagué… etc.), estos pasajeros abordan el automotor fuera de la terminal y son usuarios potenciales que completan la capacidad del bus. Pero el costo de estos viajes a ciudades intermedias no se ha establecido, no existe una unidad de medida o valor definido, en la práctica el costo que tiene un tramo de ruta de transporte terrestre intermunicipal es difícil de calcular, se ha convertido en un problema complejo para las compañías de transporte público puesto que en algunas ocasiones, se realizan cobros excesivos en los pasajes debido a que el conductor o su acompañante son quienes deciden sobre cuál es el valor que se cobrará al pasajero por la distancia que ha recorrido. Lo anterior, solo genera malestar entre los usuarios, fomenta la evasión, la competencia desleal y, por otro lado, en algunos momentos el dinero recaudado es desconocido para la compañía de transporte, puesto que solo se tiene en cuenta los pasajeros que embarcan en la terminal.

Aunque el ministerio de transporte permite la libertad de tarifas para la prestación del servicio público de transporte terrestre automotor de pasajeros por Colombia, y en su artículo 3 se explica cómo se calcula el costo de la tarifa: “las empresas de transporte mantendrán en sus archivos de estudio y las estructuras de costos elaborados directamente o a través de las entidades gremiales, que dieron origen al cálculo de las tarifas establecidas”[1].

“La estructura de los costos deberán tener en cuenta los derechos de uso de los terminales de transporte terrestre según el nivel de servicio, el seguro de pasajeros y el valor correspondiente que se destinará para la reposición de los equipos de acuerdo con el programa y el fondo de reposición”[1], el cobro por unidad o tramo recorrido en una ruta de transporte terrestre intermunicipal no está contemplado en esta norma.

En la actualidad, la tecnología asociada al transporte público ofrece dispositivos como: barras, cámaras o sensores de conteo de pasajeros, los cuales no identifican al pasajero y no calculan el valor asociado a la cantidad de kilómetros recorridos por el usuario; tampoco existen aplicaciones de software que empleen técnicas de biometría para reconocimiento de personas o gestión de tarifas. El desafío más grande desde la ingeniería es poder desarrollar un producto de software que pueda adaptarse a un ambiente dinámico.

1.2. Formulación del problema

Basado en lo anterior el problema se delimita de la siguiente manera:

¿Cómo identificar a un pasajero para calcular la tarifa que debe cancelar por los kilómetros recorridos en una ruta de transporte terrestre intermunicipal en Colombia?

De la pregunta anterior se desprenden los siguientes interrogantes:

- ¿Cómo identificar al pasajero al momento de abordar el automotor fuera de la terminal y al descender en el sitio de destino?
- ¿De qué manera se puede calcular la tarifa que debe cancelar un pasajero por los kilómetros recorridos en una ruta de transporte terrestre intermunicipal?
- ¿Cómo notificar al pasajero acerca del costo asociado a los kilómetros que ha recorrido en una ruta de transporte terrestre intermunicipal?

2. OBJETIVOS

Para el desarrollo de este trabajo de grado se han trazado los siguientes objetivos.

2.1. Objetivo general

Desarrollar un prototipo que permita identificar al pasajero para calcular la tarifa de los kilómetros recorridos en una ruta transporte terrestre intermunicipal.

2.2. Objetivos específicos

- Identificar al pasajero usando biometría al momento de abordar el automotor fuera de la terminal y al descender en el destino.
- Calcular el valor de la tarifa que debe ser cancelada por el pasajero de acuerdo con los kilómetros recorridos.
- Desarrollar un módulo de interacción con el usuario que le permita a éste conocer el valor que debe pagar.

3. ALCANCES Y LIMITACIONES

Este trabajo de grado tiene como alcance el desarrollo de un prototipo funcional experimental que permitirá identificar al pasajero, calcular la tarifa e interactuar con el usuario para informar que valor que debe cancelar por el servicio prestado.
Por lo tanto, este prototipo no podrá:

- Obtener información del usuario.
- Hacer estadísticas de viaje.
- Cotejar información con bases de datos de policía, fiscalía o interpol.
- Realizar video vigilancia.
- Controlar un historial de viajes de un usuario.
- Reconocer personas requeridas por la justicia.

Consideraciones:

- El prototipo solo podrá funcionar sobre un mini pc Raspberry pi 3 con sistema operativo Raspbian.
- El prototipo se construirá con tecnologías como: Python, OpenCV, Tkinter y utilizará una base de datos embebida SQLite.
- No se asegura despliegue en dispositivos móviles o tabletas.
- Para efectos de las pruebas como entorno de ejecución se utilizará un mini pc Rasberry pi 3. No se asegura que funcione en otros mini pc o hardware diferente.
- Para validar el prototipo se utilizarán bases de datos libres que se obtengan de la red.
- Los datos e información capturada por el prototipo estarán regulados por la política de tratamiento de datos personales.

4. JUSTIFICACIÓN

En este trabajo de grado se realizó la implementación de un prototipo hecho a la medida, que permite el cobro justo de tarifas en transporte terrestre intermunicipal y cuenta con ciertos elementos que le permiten denominarse un proyecto de grado en el marco del programa académico de maestría en ingeniería de software, dichos elementos son: el desarrollo de software, Ingeniería de requerimientos, bases de datos, procesamiento digital de imágenes, pruebas de software entre otros.

4.1. Pertinencia

- Es importante destacar el uso de la biometría en el recaudo de pasajes, debido a que en la actualidad se encuentra enfocada hacía el control, la vigilancia y la seguridad de los pasajeros.
- Este proyecto en cierta medida permitió: interoperar con barras de conteo (contadores de pasajeros) las cuales se encuentran instaladas en vehículos de transporte intermunicipal, permitiendo maximizar la productividad, diversificar las tarifas[1] e indirectamente beneficiar a las empresas de transporte terrestre intermunicipal quienes recaudaran un rubro que hasta el momento es desconocido.
- Se pretende con este prototipo funcional experimental acercar el sector del transporte a la academia, generando un vínculo entre la industria y la educación, cuyo objetivo sea aportar el mayor grado de conocimientos que permitan dar solución a múltiples problemas.
- El prototipo funcional experimental será de gran utilidad para las empresas de transporte público porque permitirá contar con información más precisa de sus usuarios, se incrementarán sus dividendos y generará un cambio en el modo de competir en el sector del transporte terrestre intermunicipal.

4.2. Viabilidad

- El uso de la biometría para el cobro de pasajes[2][3] en el sector de transporte se ha hecho muy popular desde hace más de una década; algunos estudios existentes, dan fe de que es posible utilizar las técnicas de biometría para la identificación de personas con fines de recaudo en el sector transportador.
- La práctica del grupo de investigación, el contenido curricular del programa académico de maestría en ingeniería de software y además del conocimiento adquirido por más de 10 años en desarrollo de software de quien presenta este documento, son elementos de peso para que esta propuesta se realice.

4.3. Impacto

En esta sección se explican los diferentes impactos que podría tener el desarrollo del prototipo funcional:

- Social: Los usuarios de transporte intermunicipal podrán cancelar el valor que corresponde por los kilómetros recorridos con un precio justo y acorde a su desplazamiento, evitando los cobros excesivos en las tarifas y retrasos en los desplazamientos.
- Salud: El usuario no tendrá contacto con el dispositivo puesto que se debe tomar una distancia aproximada de 50 centímetros para la captura de imágenes, evitando la contaminación por contacto.
- Económico: Las compañías de transporte público intermunicipal serán directamente beneficiadas debido a que se podrá recaudar un rubro que hasta el momento era desconocido.
- Ambiental: Permitiría reducir el uso de papel al generar tiquetes de viaje, siendo su rostro la forma de pago de pasajes. Se reduce el uso de plástico evitando la creación de tarjetas inteligentes o RFID.

5. MARCO REFERENCIAL

En esta sección se explican algunos conceptos que se utilizaron durante el desarrollo del proyecto. Contiene un marco teórico donde se explican conceptos teóricos utilizados durante la implementación. Un marco conceptual que define algunos términos muy usados en este documento.

5.1. Marco teórico

En este apartado se definen conceptos teóricos que se utilizaron en construcción del proyecto y que sirven de apoyo al lector.

5.1.1. **Reconocimiento facial**:

El reconocimiento facial es una forma precisa de identificar a una persona, sumado a esto que los rasgos conductuales o físicos de los humanos siempre están presentes sin llegar a perderse u olvidarse. Como se muestra en la Figura 1, este proceso se realiza mediante varias fases.

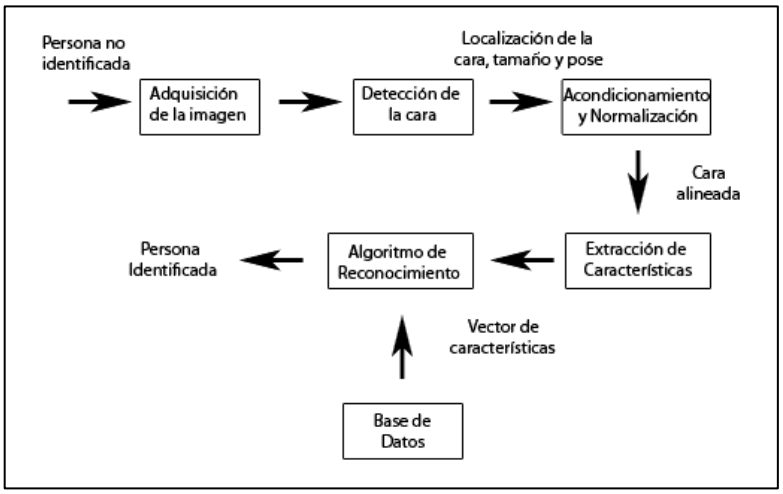


Figura 1. Proceso de reconocimiento facial[4].

Como todo proceso, inicialmente el sistema detecta el rostro del usuario y para hacerlo, el sistema utiliza un clasificador en cascada que usa funciones o medida Haar propuesto en [5] por Paul Viola y Michael Jones.

El algoritmo Viola-Jones es un método que tiene como principales características su alta velocidad en la detección de objetos y un alto porcentaje de acierto. Este alto porcentaje de acierto, se logra por un refinamiento que se hace al clasificador tradicional AdaBoost y porque iteración tras iteración se convierte en un clasificador fuerte, mejorando sustancialmente.

Este algoritmo cuenta con los siguientes pasos:

1. *Detección imágenes por características Haar*: Existen 3 tipos de características Haar estas son: de borde, lineales, diagonales. Véase Figura 2

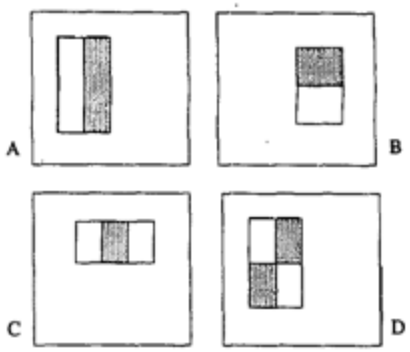


Figura 2. Tipo de características Haar y ventana de búsqueda[5].

Tal como se representa en la Figura 2, existen 3 tipos de características:

- Si hay dos rectángulos juntos (A) o (B), el valor de esta característica se calcula con la diferencia entre la suma de los pixeles de cada sección.
- Si hay tres rectángulos juntos (C), el valor de esta característica se calcula sumando de los pixeles de las secciones externas y con el resultado anterior se hace una diferencia con el valor de la suma de los pixeles de la sección del medio.
- Si hay cuatro rectángulos (D), el valor se obtiene de la diferencia entre la suma de pixeles de áreas diagonales pares[4][5].

En [5], las características se calculan sobre una matriz de búsqueda de 24x24 pixeles que en total resultan 180.000 características[7].

2. *Velocidad al calcular las características de Haar*

La rapidez en el cálculo de estas características se debe a una representación llamada imagen integral (un proceso muy similar al llamado summed área table en [8]) que en cada posición (x, y)contiene la suma de los pixeles que se encuentran arriba y a la izquierda de esa posición en la imagen original [5][6].

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y')$$

Figura 3.Cálculo de la suma de los pixeles de la imagen integral [6][5].

$ii(x, y)$ es la imagen integral e $i(x', y')$ es la imagen original, la imagen integral se calcula en un solo paso sobre la imagen original, utilizando las siguientes funciones de manera repetitiva[5][6]:

$$s(x,y) = s(x, y-1) + i(x,y)$$
$$ii(x,y) = ii(x-1, y) + s(x,y)$$

Figura 4. Fórmula para el cálculo de la imagen integral[5][6].

En Figura 4, $s(x, y)$ se refiere a la suma acumulada de la fila x, siendo $s(x, -1) = 0$ y $ii(-1, y) = 0$. El uso de la imagen integral hace posible que calcular las características de dos rectángulos se pueda obtener su valor con 6 posiciones en la matriz; las características de 3 rectángulos con 8 posiciones y por último 9 posiciones para características de cuatro rectángulos [5][6].

3. *Entrenar el clasificador AdaBoost en cascada*

Hasta este punto se tiene un conjunto reducido de características y un conjunto de entrenamiento, con estos elementos se requiere el uso de aprendizaje automático con entrenamiento supervisado.

El método Viola-Jones, utiliza el algoritmo AdaBoost (Adaptative Boosting), una versión mejorada del Boosting original[7], éste permite el uso de múltiples clasificadores “débiles” con el fin de construir un clasificador “fuerte” con una alta precisión.

Para entrenar al clasificador de las 180.000 características, se extrae un pequeño grupo de estas y a cada clasificador “débil” se asigna una característica. Este clasificador establece un umbral que reduce significativamente los ejemplos que son mal clasificados; entonces, un clasificador “débil” $h_i(x)$ consta de una característica f_j un umbral θ_j y una polaridad P_j, este último indica el signo en la desigualdad.

$$h_j(x) = \begin{cases} 1 & si\ p_j f_i(x) < p_j \theta_j \\ 0 & en\ otro\ caso \end{cases}$$

Estos son los pasos del algoritmo AdaBoost que en cada iteración selecciona una de las 180.000 posibles.

- Obtener un conjunto de imágenes ejemplo $(x_i, y_i), \ldots, (x_n, y_n)$ donde $y_i = 0{,}1$ para ejemplos negativos y positivos.
- Inicializar los pesos $w_{1i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_i = 0{,}1$ respetivamente, cuando m y l son números negativos y positivos respectivamente.
- Para $t = 1, \ldots, T$:
  a) Normalizar los pesos $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,i}}$asi pues w_t es una distribución de probabilidad.
  b) Para cada característica j, entrenar un clasificador h_j que solo use una característica. El error es evaluado con respecto a $w_t, \epsilon_j = \sum_i wi \left|h_j(x_i) - y_i\right|$
  c) Seleccione el clasificador h_t con el error más bajo ϵ_j
  d) Se actualizan los pesos $w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$ donde $e_i = 0$ si el ejemplo x_i es clasificado correctamente, $e_i = 1$ en otro caso y $\beta_t = \frac{\epsilon_j}{1-\epsilon_j}$

El clasificador fuerte final es: $h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & en\ otro\ caso \end{cases}$

donde $\alpha_t = \log \frac{1}{\beta_t}$.

Cuando se tiene el clasificador construido como se describió anteriormente, la idea es construir clasificadores más pequeños que sean capaces de descartar las ventanas negativas (es decir aquellas que no contienen al objeto buscado) mientras que se detectan más las ventanas positivas (es decir aquellas que sí contienen el objeto buscado), en este orden de ideas, se puede ajustar el umbral para que el clasificador reforzado obtenga una tasa de falsos negativos cerca a cero [5] (Véase Figura 5).

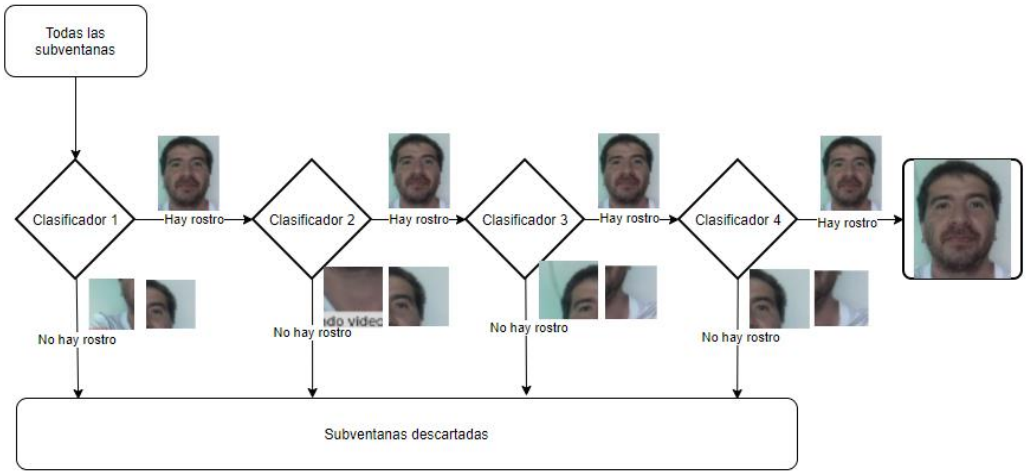


Figura 5. Funcionamiento de AdaBoost cascada. Fuente: Realización propia

La cascada completa de detección de rostros tiene 38 etapas con más de 6000 funciones; sin embargo, utiliza la mitad de 10 revisiones de características por subventana[5][6].

4. Uso del clasificador fuerte para la detección

Para reducir la variabilidad en las condiciones de luz, todas las ventanas secundarias de entrenamiento se normalizaron, de igual manera la normalización es necesaria en el proceso de detección. Ésta se debe aplicar después de calcular el valor de cada característica y no antes.

La varianza es utilizada para normalizar $\sigma^2 = m^2 - \frac{1}{N}\sum x^2$ donde m es la media que se calcula usando la imagen integral y x es el valor del píxel en la subventana. Dado que el detector final es insensible a pequeños cambios en la traducción y la escala, generalmente se producirán múltiples detecciones alrededor de cada cara en una imagen escaneada[5]. De hecho, se puede exigir que las detecciones tengan un determinado número mínimo de detecciones vecinas para disminuir el número de falsos positivos [6].

En la primera etapa, el sistema detecta el rostro del usuario y posteriormente captura una imagen que se guarda como la marca de inicio de ingreso al sistema, luego se captura una segunda imagen y con esta se procede automáticamente a realizar el proceso de reconocimiento.

Antes de iniciar el reconocimiento se debe realizar un pre-procesamiento de la imagen, esto con el fin de que extracción de características pueda ser efectiva en su totalidad.

El proceso realizado consta de capturar la imagen a cierta distancia donde la cámara pueda enfocar correctamente al usuario, con esto se desea que todas las imágenes sean capturadas de la misma forma y posición, después de realizado el proceso anterior, se procede a recortar la imagen. Dicha imagen recortada solo tiene el área del rostro del usuario. Por último, para finalizar el proceso se hace una normalización o modificación al histograma de la imagen, convirtiéndola en escala de grises, esto con el fin de dar uniformidad asegurando el color gris en todos los pixeles y en su gran mayoría iguales (Véase Figura 6).

Figura 6. Preprocesamiento de imágenes [7].

El proceso de extracción de características se basa en obtener información relevante de una imagen o fuente de video concerniente a expresiones faciales o ciertas zonas del rostro como los ojos, la boca, entre otros. De esta manera se tiene una visión global o focalizada del rostro[9]. Se utilizará un algoritmo de reconocimiento facial que usa patrones binarios locales, técnica conocida y utilizada en la actualidad.

La técnica de patrones binarios locales fue propuesta por Timo Ojala en [10], la versión estándar funciona de la siguiente manera: se divide la imagen en una cuadricula, se hace un barrido sobre esta y por cada cuadricula de 3x3 pixeles se ubica un píxel central (pivote), siendo éste la referencia, se calcula la diferencia con los píxeles vecinos con base a la siguiente fórmula:

$$LBP_{p.r} = \sum_{p=0}^{p-1} s(g_i - g_c)2^i, S(x) = \begin{cases} 1 & if\ x \geq 0 \\ 0 & en\ otro\ caso \end{cases}$$

Si g_i es más pequeño que g_c, el resultado binario del píxel se establece en 0; de lo contrario, se establece en 1. Todos los resultados se combinan para obtener un valor de 8 bits. El valor decimal del binario es la función LBP(véase Figura 7) [11].

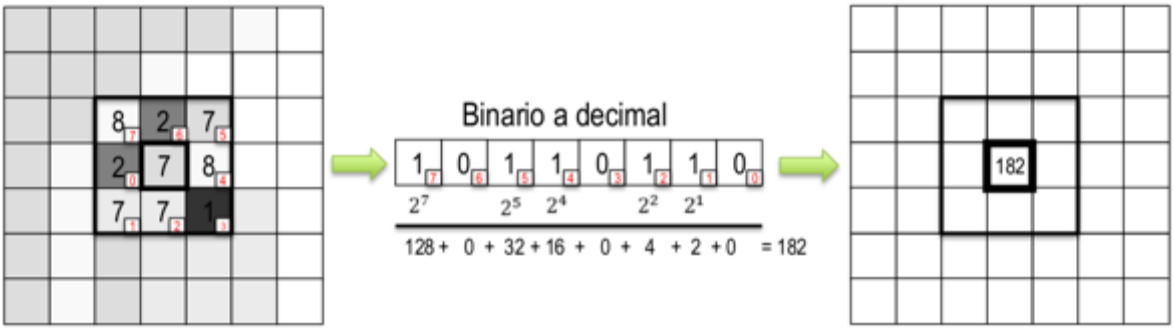


Figura 7. Operador para el patrón binario local[7].

Por ser un método de características locales, el operador LBP, posee dos atributos principales, los cuales son: soporta cambios monótonos de iluminación y la simplicidad en el cálculo [7][9].

El valor final en decimal es la etiqueta que llevará el píxel central.
El operador LBP se mejoró con el fin de acceder vecindades circulares con diferentes radios (véase Figura 8)[9]. Por lo tanto, se define la siguiente notación (P,R) lo que significa P puntos de muestreo en un círculo de radio R[7].

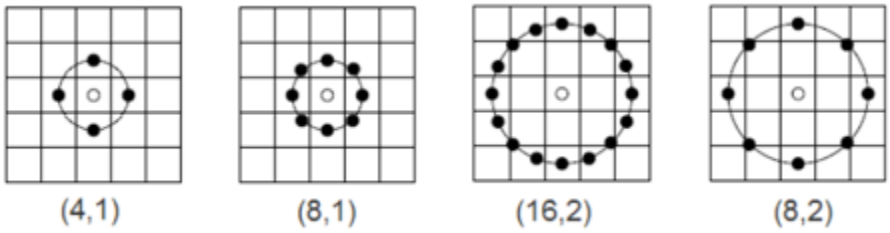


Figura 8.Diferentes radios para LBP[7].

La metodología básica para la descripción de rostros basada en LBP es la siguiente: La imagen facial se divide en regiones locales y los descriptores de textura LBP se extraen de cada región de forma independiente. Los descriptores se concatenan para formar una descripción global de la cara, como se muestra en la Figura 9[7].

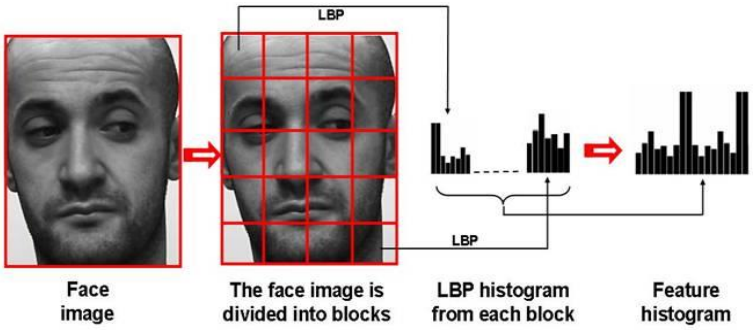


Figura 9. Descripción de rostro usando el operador LBP[7].

Finalizada la etapa anterior, se procede a identificar al usuario utilizando una imagen capturada y calculando la distancia o que tan parecida es con otras imágenes que existen en una base datos. La distancia se calcula utilizando la fórmula de la distancia euclidiana. $d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$[9].
El resultado de la distancia es un valor que indica que tan parecida es la imagen capturada con algunas de las imágenes de la base de datos, cuanto más pequeño es este valor, mejor se identifica al individuo.

5.1.2. Huella dactilar

La huella dactilar permite la identificación única de cada ser humano, se encuentra presente desde el embrión y perdura durante toda la vida[12]. Estudios afirman que, se crean por las tensiones sufridas por los dedos durante la gestación [13].
Francis Galton, en su libro FingerPrints publicado en 1892 describió las leyes de la dactiloscopia: perennidad, inmutabilidad y diversidad infinita [14][15].
En [13] se describen como:

- Perennidad: Una huella permanece intacta a lo largo de la vida de la persona con algunas excepciones (accidentes o cortes graves).
- Inmutabilidad: Una huella no se puede modificar.
- Diversidad infinita: No existen dos huellas iguales.

En una impresión dactilar se pueden observar gran cantidad de líneas o relieves[14] que se conocen como *crestas*[13] y espacios en blanco o hendiduras[14] llamados también *surcos*[13] (Véase Figura 10).

Figura 10. Huella dactilar[16].

Figura 11. Impresión huella dactilar[17].

Una característica importante para destacar, las minutas son las variaciones que pueden tener las crestas dentro de la huella y que permiten crear discontinuidad por ejemplo una terminación abrupta de una cresta o la bifurcación. Esta característica, permite a los investigadores y máquinas validar si una huella pertenece o no a una persona, debido a que estos cambios permanecen durante toda la vida, con algunas excepciones [13][18].

La captura de huellas a través de un sensor se realiza colocando el dedo sobre un superficie transparente y un sensor electrónico que se encarga de capturar la imagen, dependiendo del funcionamiento estos se clasifican en: sensores ópticos, de estado sólido, ultrasónicos[12].

Los sensores ópticos basados en reflexión de luz sobre la yema del dedo son sensores basados en *FTIR* (Transformada de Fourier Infrarroja en español): Esta técnica es la más antigua y utilizada en una gran cantidad de dispositivos. Cuando se apoya el dedo sobre un cristal que posee el sensor o también llamado prisma, un diodo led proyecta una luz difusa a través del cristal. En los valles se da una reflexión total mientras que en las crestas se dispersa [13][12]. los cambios de luz

son los que determinan el color de cada píxel (véase Figura 12). Las crestas son de color negro mientras los surcos son blancos. La calidad de la imagen obtenida depende de las condiciones de piel de cada individuo, por ejemplo factores como la sudoración o suciedad [14] (véase Figura 14).

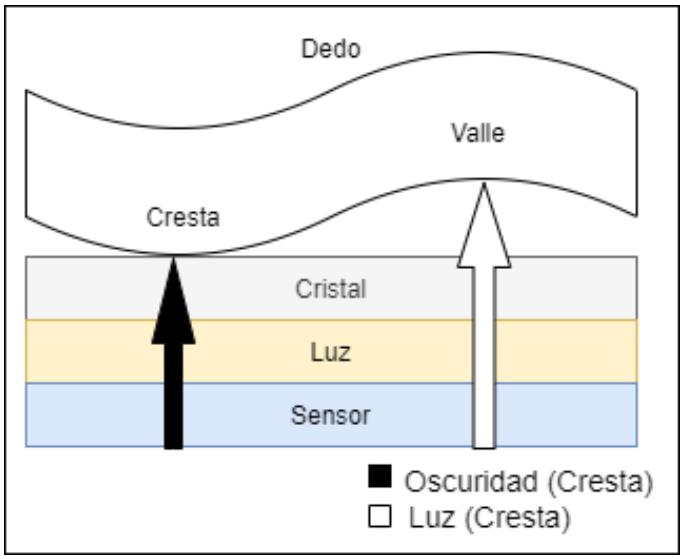


Figura 12. Funcionamiento de un sensor óptico de huella dactilar [14].

Además de capturar la huella, algunos de estos dispositivos contienen componentes como:

- **Extractor de características**: Es un componente del dispositivo que se encarga de extraer los puntos o regiones características, minutas o patrones.
- **Templates**: Componente encargado de normalizar las imágenes. se refiere a la forma como almacena las huellas.
- **Base de datos**: Componente del dispositivo que se encarga de almacenar las plantillas de las huellas. Es común que sea un espacio en memoria que puede ser o no volátil.
- **Comparador de características**: Componente encargado de comparar una huella ingresada con la base de datos de plantillas guardadas, obteniendo un resultado positivo o negativo.

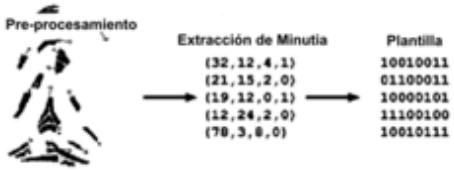


Figura 13. Proceso de comparación de huellas

Para obtener una coincidencia, el sistema del lector no necesita encontrar el patrón entero de minutas en la muestra y en la imagen almacenada, simplemente debe encontrar un número suficiente de patrones de minutas que ambas imágenes tengan en común. El número exacto varía de acuerdo a la programación del lector (véase Figura 13) [18].

Figura 14. Sensor óptico.

5.1.3. Cálculo de la distancia

Existen varias fórmulas para el cálculo de distancias, sin embargo, la más utilizada es la fórmula del semiverseno que calcula una distancia teniendo dos puntos latitud y longitud. Según [19] es una ecuación muy utilizada en navegación astronómica. Por lo general, el resultado que se obtiene de estas fórmulas es una estimación aproximada que en muchas ocasiones no se acerca a una distancia real terrestre.

$$d = 2r\ \sin^{-1}\left(\sqrt{hav(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\ hav(\lambda_2 - \lambda_1)}\right)$$

$$2r\ \sin^{-1}\left(\sqrt{sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\ sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Figura 15. Fórmula del semiverseno[20][21].

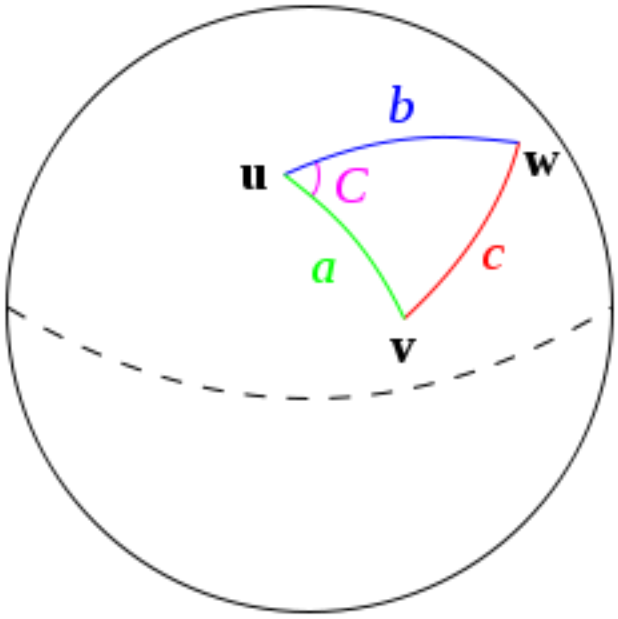


Figura 16. Triangulo esférico resuelto por la ley del semiverseno[20][21].

5.1.4. Antecedentes

En esta sección se presentan algunas investigaciones o elementos que anteceden a esta propuesta, inicialmente se citarán algunos estudios, se explicará en que consistieron y que tan parecidos son con la propuesta que presentamos en este documento.

El transporte público en muchos países de Latinoamérica al igual que en Colombia tiene grandes retos, entre ellos mejorar el servicio, cumplir con las frecuencias, evitar el sobrecupo, la seguridad al interior de los buses y el cobro de una tarifa justa. En Colombia el servicio de transporte público intermunicipal tiene libertad de

tarifas por tanto las empresas, se encargan de establecer una tabla de tarifas de acuerdo con ciertas condiciones del mercado.

En la actualidad existen dispositivos de conteo que permiten calcular la tarifa del pasajero, pero este dispositivo en ocasiones no es preciso, podría fallar calculando la tarifa o alterar el conteo de pasajeros.

Así pues, la utilización de biometría en este contexto hace más preciso el cálculo de la tarifa que el pasajero debe cancelar. El uso de estas técnicas desde hace algunos años se ha tratado de aplicar en el sector de transporte, en su gran mayoría para la seguridad de los pasajeros dentro y fuera de los buses, pero en menor medida en el cobro de tarifas o pasajes.

A continuación, se hará un breve resumen de algunos antecedentes relevantes:

- **Sistema de gestión de pasajeros basado en Reconocimiento facial para vehículos de transporte inteligentes**[2]:El método AdaBoost que utiliza el histograma de patrón binario local (LBP) se emplea para el sistema de detección de rostros puesto que utiliza un método de extracción de características robusto en cuanto a la iluminación, profundidad y enfoque.

  La rapidez al momento de hacer un reconocimiento facial en un ambiente muy dinámico es un factor importante y aunque existen muchos métodos para el reconocimiento facial, en se utilizó una estructura de clasificadores en cascada.

  El sistema desarrollado en [2] consiste en gestionar el ingreso y salida de los estudiantes en un bus escolar, en este contexto el sistema detecta los rostros de los estudiantes al ingresar y al descender del bus, posteriormente imagen detectada es enviada al servidor a través de una red inalámbrica, luego en el servidor se hace el reconocimiento facial, pues en éste ya existe una base de datos con información de los estudiantes incluyendo su foto, sumado a lo anterior, se guarda el tiempo de ingreso y salida del estudiante. La información del sistema permite a los padres conocer información como: ubicación y hora de embarque.

Para esta propuesta, se destaca el algoritmo Ada boost utilizado en [2], éste tiene un desempeño excelente en ciertas condiciones donde el contexto de la captura de imágenes es muy dinámico y podría tener problemas de luminosidad.

- **Prototipo de sistema basado en FPGA para la ubicación del rostro y el registro de personas que abordan un autobús de transporte público**[22]
  En [22] se enfoca en la reducción y prevención del crimen en el transporte público en México. Se desarrolló un dispositivo que utiliza una matriz de puertas programables FPGA (field-programmable gate array) con una cámara y con unidad de almacenamiento (Véase Figura 17).
  Datos importantes:
  La unidad FPGA se utiliza para procesar las imágenes.
  Las imágenes se almacenan en una SD.
  El formato escogido para las imágenes es BMP.

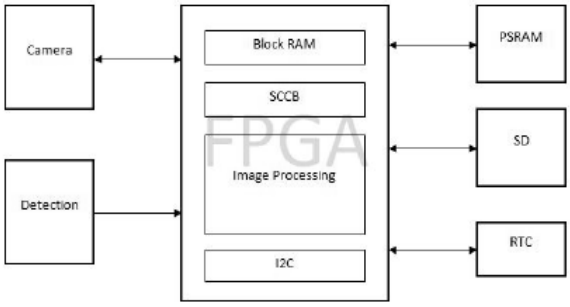


Figura 17. Diagrama de bloques.[22]

El sistema por medio de una cámara digital toma la foto de la persona y la almacena con el fin asegurar el registro y evitar que se destruya la cámara o se intente sabotear el sistema, luego se procesa utilizando matrices binarias con el fin de representar las facciones del rostro y establecer si alguna región de la imagen es o no candidatas a ser procesadas.

Las pruebas se hicieron en un bus que estuvo detenido, con una cámara de 1.3 mpx, (la cual ubicó en lugar estratégico para evitar problemas de luminosidad) y una SD. Todas las imágenes se tomaron en escala de grises y se guardaron en formato BMP. Durante la prueba el módulo FPGA solo procesó la región correspondiente al rostro ahorrando espacio de almacenamiento y mejorando el tiempo de identificación.

Las aplicaciones de este prototipo son múltiples, puede ser útil en eventos de hurto o búsqueda de persona involucradas en delitos que pudieran abordar un bus, siendo una herramienta de apoyo para la policía.
Como método para asegurar que no se presenten fraudes al ingresar al automotor, en esta propuesta, es interesante el proceso de guardar la imagen como se hace en [22].

- **Sistema de Gestión de Tarifas para Corporación de Transporte usando reconocimiento facial basado en Análisis de componentes principales**[3]**:** utiliza el reconocimiento facial para identificar al pasajero, establecer la tarifa que se cobrará, debitar de una cuenta bancaria el monto de la tarifa y finalmente informar al pasajero del cobro. El sistema captura la imagen del pasajero cuando éste se encuentra dentro del bus, la imagen de procesa y se envía a un servidor, donde por medio de un base de datos de imágenes se hace el proceso de identificación, comparando la imagen tomada con la imagen que se guardó al momento de crear su cuenta bancaria, adicionalmente se captura la ubicación donde se subió y el destino del pasajero por medio de un sensor GPS, con los datos anteriores de calcula el valor que debe cancelar el pasajero y se debita de la cuenta. Datos importantes:
    - La transmisión de los datos al servidor se hace manera inalámbrica.

- El algoritmo utilizado para la detección de rostros es el algoritmo Viola-Jones, es muy rápido en detección.
- El algoritmo basado en PCA utilizado en el reconocimiento facial.
- El sistema se implementó en Matlab.
- La imagen se toma con un vehículo en movimiento.
- Utiliza dos cámaras una de entrada y otra salida.

La investigación concluye destacando las bondades de la implementación como son el no uso de tarjetas de contacto o con tecnología RFID y la precisión con la que se realiza el reconocimiento facial.

Cabe destacar que [3] es la más aproximada a la propuesta que se plantea en este documento con algunas diferencias:

- El software Matlab es alto costo de ahí que no es posible implementar la propuesta en esta aplicación.
- Por razones de carácter socioeconómico. La población objetivo, en su gran mayoría no posee una cuenta bancaria; Esto haría que [3] no fuera viable en este contexto, debido a que la gran mayoría de usuarios son de estratos 1, 2,3 y la relación con el sistema financiero es poca o casi nula.
- El proceso de identificar al pasajero -reconocimiento facial explicado en la investigación- no es posible realizar en un servidor remoto por las condiciones geográficas que poseemos en este contexto y los múltiples problemas que existen con la conexión a internet a través de redes inalámbricas, datos celulares o gprs, algo que imposibilita de sobre manera el envío de información remota.
- Esta propuesta está enfocada a mejorar el servicio y realizar un cobro justo de la tarifa que se cobrará al pasajero por lo que ha recorrido y no pretende descartar otros métodos de identificación o recaudo.
- En la literatura se encuentran otros documentos que no corresponden al contexto donde se desarrollará la propuesta, pero que si aportan

conocimiento. En esta sección se hace un breve recuento de los más destacados:

Tabla 1. Resumen otros antecedentes.

Título	Descripción
Sistema de reconocimiento de eventos de audio vídeo para la seguridad del transporte público.[23]	Se desarrolló un sistema de video vigilancia que sirve de apoyo a la policía para esclarecer delitos dentro de los buses de transporte público. Éste consta de seis módulos, incluidos en particular tres nuevos: (i) Detección y seguimiento de rostros, (ii) Detección de eventos de audio y (iii) Reconocimiento de escenarios de audio y video. En general el sistema permite con un módulo de detección y seguimiento de rostros, rastrear las personas y realizar seguimiento; otro modulo importante es aquel de detección de eventos que permite por medio de un micrófono capturar sonidos poco convencionales.
Smart E-Ticketing System para autobuses de transporte público.[24]	Se desarrolló un sistema inteligente que Asigna automáticamente el asiento al pasajero, generando y enviando el boleto digital a la cuenta de correo del usuario y al tiempo debitar el pago de un valor recargado previamente, promoviendo la digitalización del dinero y fomentando la iniciativa de ciudad inteligente.
Sistema de reconocimiento de señales de tránsito basado en una red neuronal convolucional[25]	Se desarrolló un sistema que toma una imagen digital de las señales de tráfico y a través de una red neuronal convolucionar identifica y la señal y emite una alerta con base al significado de la ésta.

Finalizando con los antecedentes, en esta sección se presentan en la actualidad que se ha desarrollado o implementado y que se encuentra relacionado con esta propuesta.

- La compañía opto control, se destaca entre sus productos modelo OPTR-5000, un contador de personas que permite calcular la tarifa que debe cancelar el pasajero a través de una unidad GPS instalada dentro del dispositivo, con ello captura la ubicación del pasajero al momento de abordar y descender del automotor, con base a estos datos, calcula el costo que debe cancelar[26].
- La empresa de transporte Shanghai Jiushi Bus Company de la ciudad de Shangai en china implementaría cámaras con reconocimiento facial para conductores[27].
  El sistema garantizaría el estado anímico de los conductores por medio de las expresiones de su rostro y detectar conductas irregulares como hablar por celular o fumar. Ésta información se enviaría a la central con el fin de tomar medidas en contra del conductor[27].
- Transmilenio el sistema de transporte masivo, de Bogotá, implementó un sistema de reconocimiento facial con el fin de rastrear y capturar delincuentes que ingresan al sistema. El sistema de reconocimiento facial captura millones de rostros en segundos, almacenando esta información en una central la cual se puede acceder desde otros dispositivos como celulares, tabletas, permitiendo a los agentes en cada estación estar alerta.

5.1.5. Trabajos relacionados

En esta sección se relacionan documentos o implementaciones que pueden llegar a ser similares a esta propuesta.

Institución	Universidad Tecnológica Nacional
País	Argentina
Año	2018
Descripción	“El sistema se basa en el procesamiento de imágenes en tiempo real para monitorear las acciones del conductor, mediante la detección de parpadeo y la posición de la cabeza. Se ha utilizado la herramienta OpenCV para detectar de forma exacta el ojo y calcular su relación de aspecto. Con esta información, se puede determinar apertura o no del ojo, si el mismo persiste cerrado por un tiempo critico indica somnolencia. Si esto acontece, el sistema alertará al conductor.[28]”.

Institución	Escuela superior politécnica de Chimborazo
País	Ecuador
Año	2018

Descripción	“El objetivo del presente trabajo de titulación fue de diseñar e implementar una plataforma de pagos de servicios de transporte público usando smartphones con soporte de NFC. Se estudiaron las arquitecturas que posee la tecnología NFC y las tramas del protocolo de comunicación para la creación de aplicaciones tanto de escritorio como móviles que operarán en smartphones compatibles con Android, las mismas que fueron desarrolladas en Java. De igual manera, se creó una base de datos en MySQL, alojada en un servidor, donde se guardarán todos los datos de usuarios, historiales y transacciones que realice la plataforma para control y administración del sistema[29]”.

Institución	Universidad regional autónoma de los andes
País	Ecuador
Año	2019
Descripción	“La presente investigación tuvo como finalidad desarrollar un sistema informático de reconocimiento facial para el registro y control de asistencia de los socios de la cooperativa de Taxis y Camionetas Puyo, que permita llevar un control adecuado del registro de las asistencias de los socios a reuniones y convocatorias dentro de la sede de la Cooperativa[30]”.

Institución	SmartBus: Big Data & Data Science en Transporte Urbano
País	España
Año	2017
Descripción	“El sistema se basa en una plataforma Big Data mediante la cual se almacenar grandes cantidades de datos procedentes de diversas fuentes, tanto internas a la EMT como externas, con el fin de aplicar técnicas de aprendizaje automático para segmentar las líneas de autobús y posteriormente realizar una predicción de la demanda diaria en una de ellas[31]”.

Institución	Pontificia universidad católica de Valparaíso
País	Chile
Año	2012
Descripción	“El objetivo de este proyecto es resolver el problema de los dueños de microbuses de la línea “Asociación de buses de San Antonio”, el cual consiste en no tener una manera de conocer la cantidad exacta de dinero recaudado por sus máquinas al final de una jornada laboral. Así surge la necesidad de crear un sistema que cumpla esta función; utilizando tecnología RFID (Identificación por radiofrecuencia)

se diseñará una solución al problema expuesto y además se incorporarán funcionalidades extras que ayuden al usuario a la toma de decisiones relacionadas con el negocio[49].”

5.2. Marco conceptual

En esta sección se definen componentes hardware y algunas tecnologías de software que se utilizaron en la construcción del proyecto.

5.2.1. Hardware

En este apartado se describe la plataforma de hardware sobre la cual se ejecutará el software y algunos periféricos

La fundación Raspberry pi es una organización benéfica ubicada en el Reino Unido, su función es fomentar el uso de la informática en todas las personas del mundo, distribuyendo computadoras de bajo costo y alto rendimiento incentivando la educación [32].

5.2.1.1. **Rasberry pi**

Es un microcomputador portátil, muy económico y con alto rendimiento, tiene el tamaño de una tarjeta de crédito y posee casi todas las prestaciones de un computador portátil tanto en hardware como en software [33].
En el desarrollo del proyecto se utilizaron diferentes modelos de Raspberry pi, descritas a continuación:

Tabla 2. Especificaciones técnicas Raspberry Pi 3 Modelo B[34]

Modelo B

Figura 18. Raspberry Pi 3, Modelo B[34]

Procesador	Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
Memoria	1GB RAM
Conectividad	BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
	100 Base Ethernet
Puertos	40-pin extended GPIO
	4 USB 2 ports
Video y audio	4 Pole stereo output and composite video port
	Full size HDMI
Conexión externa	CSI camera port for connecting a Raspberry Pi camera
	DSI display port for connecting a Raspberry Pi touchscreen display
Almacenamiento	Micro SD port for loading your operating system and storing data
Alimentación	Upgraded switched Micro USB power source up to 2.5A

Tabla 3. Especificaciones técnicas Raspberry Pi · Modelo B+[35]

Modelo B+	
	
Figura 19. Raspberry Pi, Modelo B+[35]	
Procesador	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
Memoria	1GB LPDDR2 SDRAM
Conectividad	2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE

	Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
Puertos	Extended 40-pin GPIO header
	Full-size HDMI
Video y audio	4 USB 2.0 ports
	CSI camera port for connecting a Raspberry Pi camera
Conexión externa	DSI display port for connecting a Raspberry Pi touchscreen display
	4-pole stereo output and composite video port
Almacenamiento	Micro SD port for loading your operating system and storing data
Alimentación	5V/2.5A DC power input
	Power-over-Ethernet (PoE) support (requires separate PoE HAT)

Tabla 4. Especificaciones técnicas Raspberry Pi 4 Modelo B[36]

Pi 4 Modelo B	
	
Figura 20. Raspberry pi 4, modelo B[36]	
Procesador	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memoria	1GB, 2GB or 4GB LPDDR4-3200 SDRAM (depending on model)
Conectividad	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
	Gigabit Ethernet
Puertos	2 USB 3.0 ports; 2 USB 2.0 ports.
	Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
Video	2 × micro-HDMI ports (up to 4kp60 supported)
	2-lane MIPI DSI display port
Conexión externa	2-lane MIPI CSI camera port
Audio	4-pole stereo audio and composite video port
Almacenamiento	H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
Alimentación	Micro-SD card slot for loading operating system and data storage
	5V DC via USB-C connector (minimum 3A*)
	5V DC via GPIO header (minimum 3A*)

Periféricos: Adicionalmente a la placa base se requieren otros dispositivos como: cámaras, lectores de huella digital entre otros. A continuación, se describe las características de cada uno.

5.2.1.2. Cámara

El sistema requiere de una cámara para la captura de imágenes y considerando las especificaciones de la placa base, se utilizaron dos tipos:

Cámara 1: Es un módulo de cámara conectada directamente sobre la plataforma de hardware a través de la interfaz CSI. En la siguiente tabla se describen sus especificaciones:

Tabla 5. Especificaciones técnicas cámara 1

Módulo de Cámara	
 *Figura 21. Módulo de cámara*[37]	
Resolución	imágenes estáticas de 2592 x 1944 píxeles y también soporta 1080 p @ 30 fps, 720 p @ 60 fps y grabación de vídeo de 640 x 480 p 60/90, con sensor de cámara web de 5 MP OV5647 1080p.
Interfaz de conexión	CSI
Tecnología de enfoque	Manual
Tecnología de lente	Estándar
Micrófono integrado	No
Visión nocturna	si
Campo visual	50º
Interfaz de conexión	CSI

Cámara 2: Es una cámara web convencional que se conecta al microcomputador a través de la interfaz bus serie universal (USB). En la siguiente tabla se describen sus especificaciones.

Tabla 6. Especificaciones técnicas cámara 2

Cámara WEB

Figura 22. Cámara Logitech[38]

Resolución máxima	720p/30 fps
Tipo de enfoque:	Automático
Tecnología de lente	Estándar
Micrófono integrado	No
Campo visual:	69º
Interfaz de conexión	USB
Visión nocturna	No

5.2.1.3. Lector de huella

Este periférico está considerado opcional en el sistema, permite capturar la huella dactilar del usuario. En la siguiente tabla se describen sus especificaciones.

Tabla 7. Especificaciones técnicas lector de huella

Módulo lector de huella	
	
Figura 23. Módulo de huella dactilar[39]	
Voltaje:	3.3V
Resolución:	CSI
Color de luz	Si
Interfaz	Serie TTL
Corriente máxima	60mA

Tiempo de lectura de huella digital	<1.0
Dimensiones	48x21x24

5.2.1.4. Módulo GPS

Este periférico permite al sistema obtener la ubicación inicial y final de un usuario, insumo indispensable para calcular la distancia recorrida y la tarifa. En la siguiente tabla se describen sus especificaciones.

Tabla 8. Especificaciones técnicas módulo GPS

Módulo GPS	
 Figura 24. Módulo GPS[40]	
Voltaje:	3-5V
Tipo antena	cerámica
Memoria	EEPROM para guardar datos de configuración cuando el módulo se desenergice
Interfaz	Serial UART 5V
Frecuencia de refresco:	5Hz
Batería de respaldo	(MS621FE)
Tamaño de la antena:	25mm x 25mm
Tamaño del módulo:	25mm x 35mm
Indicador de señal	LED
Baud rate por defecto:	9600bps
Soporta	SBAS (WAAS, EGNOS, MSAS, GAGAN)

5.2.2. Software

A continuación, se describen las tecnologías de software utilizadas en el desarrollo del proyecto:

5.2.2.1. **Raspbian**

Sistema operativo optimizado para Raspberry pi, basado en Debian que contiene más de 35.000 paquetes compilados, entre los que se incluyen lenguajes de programación y herramientas de desarrollo.

Aunque Mike Thompson (mpthompson) y Peter Green (plugwash) son independientes, la fundación Raspberry siempre ha estado muy agradecida por el gran aporte que han hecho a sus dispositivos, siendo Raspbian el más optimizado y utilizado[41].

5.2.2.2. **Python 3.7.2**

Es un lenguaje de programación de alto nivel orientado a objetos, permite el desarrollo rápido de aplicaciones. El lenguaje de programación no tiene pasos de compilación por ser un lenguaje script, es muy rápido en ejecución, posee un depurador que muy efectivo[42][43].
Cuando se desarrolla en Python se crean módulos que permiten ser reutilizados en otros proyectos, con la ventaja del ahorro de tiempo y esfuerzo [43][42].
Está orientado al desarrollo web y móviles, es gratuito al igual que un sinnúmero de bibliotecas que se pueden utilizar en múltiples plataformas. Cabe destacar el uso de este lenguaje en el ámbito del procesamiento de imágenes y video, integrando bibliotecas como OpenCV, para el reconocimiento facial o análisis de imágenes en tiempo real [42][43].

5.2.2.2.1. Tkinter

Tk / Tcl ha sido durante mucho tiempo una parte integral de Python. Proporciona un kit de herramientas de ventanas robusto e independiente de la plataforma, que está disponible para los programadores de Python que utilizan el paquete Tkinter y su extensión, Los módulos tkinter.tix y tkinter.ttk.

Tkinter es un conjunto de envoltorios que implementan los widgets Tk como clases de Python. Además, el módulo interno _tkinter proporciona un mecanismo seguro para subprocesos que permite que Python y Tcl interactúen[44].

5.2.2.2.2. PyFingerprint

La biblioteca PyFingerprint permite utilizar los sensores de huellas digitales ZhianTec ZFM-20, ZFM-60, ZFM-70 y ZFM-100 en la Raspberry Pi u otras máquinas Linux. Algunos otros modelos como R302, R303, R305, R306, R307, R551 y FPM10A también están soportados[45].

5.2.2.3. **OpenCV**

(Open Source Computer Vision Library) es una biblioteca de software de aprendizaje por computadora y visión por computadora de código abierto. OpenCV se creó para proporcionar una infraestructura común para aplicaciones de visión por computadora y para acelerar el uso de la percepción de máquinas en los productos comerciales. Al ser un producto con licencia BSD, OpenCV facilita a las empresas utilizar y modificar el código. [46]
Con 2500 algoritmos optimizados, permite la detección y reconocimiento de objetos, personas, rostros, entre otros procesos de visión computacional y aprendizaje automático[46].

OpenCV es multiplataforma, permite instalarse en múltiples sistemas operativos como Windows, Linux, Mac, también en Raspbian o Noobs (sistemas operativos para procesadores ARM). Además, se puede utilizar OpenCV en lenguajes de programación como Python y Java, entre otros.

5.2.2.4. **SQLite**

Es una biblioteca escrita en lenguaje C que implementa un motor de base de datos SQL transaccional autónomo, sin servidor y de cero configuraciones. El

código para SQLite es de dominio público y, por lo tanto, es de uso gratuito para cualquier propósito, comercial o privado.

SQLite es un motor de base de datos SQL incorporado. A diferencia de la mayoría de las otras bases de datos SQL, SQLite no tiene un proceso de servidor separado. SQLite lee y escribe directamente en archivos de disco normales y ofrece una base de datos SQL completa con múltiples tablas, índices, disparadores y vistas contenidas en un solo archivo de disco. El formato de archivo de la base de datos es multiplataforma: puede copiar libremente una base de datos entre sistemas de 32 bits y 64 bits o entre arquitecturas Big-Endian y Little-Endian. Estas características hacen de SQLite, una opción popular como formato de archivo de aplicación[47].

5.2.3. Herramientas

En el desarrollo de la aplicación fue necesario el uso de un entorno de desarrollo que permitiera conexión remota a través de una red LAN, por esta razón se utilizó Visual Studio Code[48]. Un entorno integrado de desarrollo multiplataforma y multilenguaje desarrollado por Microsoft, con licencia de uso libre.

Figura 25. Logotipo de Visual Studio Code[48]

6. ANÁLISIS

En este capítulo se describe el proceso realizado para la obtención, refinamiento y especificación de los requisitos. Se inicia con la descripción de los requisitos y posteriormente su especificación.

6.1. Requerimientos funcionales

En esta sección se realiza la primera etapa de obtención de requisitos:

1. La aplicación debe permitir capturar por medio de la cámara una imagen de 640x480 pixeles del usuario.
2. La aplicación debe permitir procesar la primera imagen de 640x480 píxeles iniciando el proceso de detección.
3. La aplicación debe permitir guardar la primera imagen.
4. La aplicación debe permitir capturar por medio de la cámara una segunda imagen de 640x480 pixeles del usuario.
5. La aplicación debe permitir procesar la segunda imagen de 640x480 píxeles iniciando el proceso de detección.
6. La aplicación debe permitir guardar la segunda imagen.
7. La aplicación debe permitir identificar al usuario utilizando técnicas de biometría e imágenes capturadas previamente.
8. La aplicación debe permitir calcular la distancia recorrida por el usuario.
9. La aplicación debe permitir calcular la tarifa que el usuario debe cancelar.
10. La aplicación debe notificar al usuario el valor de la tarifa que debe cancelar por el servicio prestado.

6.2. Especificación de requisitos

En este apartado, se realiza la especificación de cada uno de los requisitos definidos en la sección 6.1.

Tabla 9. Requisito funcional 1 captura imagen 1

Tipo:	Funcional

Módulo:	Captura
• **Capturar primera imagen**	
Descripción	La aplicación debe permitir capturar por medio de la cámara una imagen de 640x480 píxeles del usuario
Entradas	
Salidas	Notificar al usuario de la realización del proceso.
Fecha de captura:	05/08/2019
Fecha de modificación	28/05/2020
Flujo del sistema	▪ El usuario debe ubicarse frente a la cámara y esperar de 1 a 3 segundos mientras el sistema realiza la captura.
Dependencias con:	

Revisado por:	**Realizado por:**
	JAIR HERNANDO VIDAL

Tabla 10. Requisito funcional 2 Procesar imagen 1

Tipo:	Funcional
Módulo:	Captura
• **Procesar primera imagen**	
Descripción	La aplicación debe permitir procesar la primera imagen de 640x480 píxeles iniciando el proceso de detección.
Entradas	
Salidas	Notificar al usuario de la realización del proceso.
Fecha de captura:	05/08/2019
Fecha de modificación	28/05/2020
Flujo del sistema	▪ El sistema debe detectar en la primera captura de imagen si existe un rostro utilizando técnicas de biometría u otras para detección de rostros y una vez haya sido detectado se procederá a guardar la

	captura. **Condiciones**: Si la imagen no contiene un rostro se descarta y se procede a realizar otra captura.
Dependencias con:	TASS-RF1
Revisado por:	**Realizado por:**
	JAIR HERNANDO VIDAL

Tabla 11. Requisito funcional 3 Guardar imagen 1

Tipo:	Funcional
Módulo:	Captura
• **Guardar primera imagen**	
Descripción	La aplicación debe permitir guardar la primera imagen.
Entradas	
Salidas	Notificar al usuario de la realización del proceso.
Fecha de captura:	05/08/2019
Fecha de modificación	28/05/2020
Flujo del sistema	▪ El sistema debe guardar la captura de la primera imagen previamente analizada y guardarla de manera temporal bajo las siguientes condiciones: o El sistema creará una subcarpeta para cada usuario que ingrese; el nombre de la carpeta es: user_[código_generado], el [código_generado] corresponde a una cadena de 8 caracteres alfanuméricos únicos generado por el lenguaje de programación. o Cada una de las imágenes capturadas, tendrán por nombre de archivo un solo número, es decir: la primera imagen tendrá por nombre 1.png donde el número (1) significa que es la imagen inicial y (png) el tipo de imagen capturada. ▪ El tamaño de la imagen que se guardará puede variar dependiendo de la distancia, profundidad entre otros factores, pero puede ser desde

	350x360 pixeles en adelante sin superar un tamaño de 400x400. ▪ La imagen que se guarda solo debe contener el área del rostro del usuario. **Condiciones**: • La imagen solo se guarda de forma temporal en el dispositivo. • El sistema debe informar al usuario cuando inicia y termina el proceso de captura de la imagen.
Dependencias con:	TASS-RF2
Revisado por:	**Realizado por:**
	JAIR HERNANDO VIDAL

Tabla 12. Requisito funcional 4 Captura de imagen 2

Tipo:	Funcional
Módulo:	Captura
• **Capturar segunda imagen**	
Descripción	La aplicación debe permitir capturar por medio de la cámara una segunda imagen de 640x480 píxeles del usuario.
Entradas	
Salidas	Notificar al usuario de la realización del proceso.
Fecha de captura:	05/08/2019
Fecha de modificación	28/05/2020
Flujo del sistema	▪ El usuario debe ubicarse frente a la cámara y esperar de 1 a 3 segundos mientras el sistema realiza la captura.
Dependencias con:	
Revisado por:	**Realizado por:**
	JAIR HERNANDO VIDAL

Tabla 13. Requisito funcional 5 Procesar imagen 2

Tipo:	Funcional
Módulo:	Captura
• **Procesar segunda imagen**	
Descripción	La aplicación debe permitir procesar la segunda imagen de 640x480 píxeles iniciando el proceso de detección.
Entradas	
Salidas	Notificar al usuario de la realización del proceso.
Fecha de captura:	05/08/2019
Fecha de modificación	28/05/2020
Flujo del sistema	▪ El sistema debe detectar en la segunda captura de imagen si existe un rostro utilizando técnicas de biometría u otras, para detección de rostros y una vez haya sido detectado se procederá a guardar la captura. **Condiciones**: • Si la imagen no contiene un rostro se descarta y se procede a realizar otra captura.
Dependencias con:	TASS-RF1
Revisado por:	**Realizado por:**
	JAIR HERNANDO VIDAL

Tabla 14. Requisito funcional 5 Guardar imagen 2

Tipo:	Funcional
Módulo:	Captura
• **Guardar segunda imagen**	
Descripción	La aplicación debe permitir guardar la segunda imagen.
Entradas	
Salidas	Notificar al usuario de la realización del proceso.
Fecha de captura:	05/08/2019
Fecha de modificación	28/05/2020 15/06/2020
Flujo del sistema	▪ El sistema debe guardar la captura de la segunda imagen previamente analizada y guardarla de manera temporal bajo las siguientes condiciones: o El sistema utilizará la misma subcarpeta creada al usuario en el momento de capturar la primera imagen. o Cada una de las imágenes capturadas, tendrán por nombre de archivo un solo número, es decir: la segunda imagen tendrá por nombre 2.png donde el número (2) significa que es la imagen secundaria y (png) el tipo de imagen capturada. ▪ El tamaño de la imagen que se guardará puede variar dependiendo de la distancia, profundidad entre otros factores, pero puede ser desde 350x360 pixeles en adelante sin superar un tamaño de 400x400. ▪ La imagen que se guarda solo debe contener el área del rostro del usuario. **Condiciones**: • La imagen solo se guarda de forma temporal en el dispositivo • El sistema debe informar al usuario cuando inicia y termina el proceso de captura de la imagen.
Dependencias con:	

Revisado por:	**Realizado por:**
	JAIR HERNANDO VIDAL

Tabla 15. Requisito funcional 7 identificación de usuario.

Tipo:	Funcional
Módulo:	Identificar
• **Identificación de usuario**	
Descripción	La aplicación debe permitir identificar al usuario utilizando la segunda imagen capturada y por medio de reconocimiento facial. Realizar una comparación con la primera imagen capturada del mismo usuario, el resultado de esta operación nos indicará un porcentaje de igualdad.
Entradas	
Salidas	Notificar al usuario de la realización del proceso.
Fecha de captura:	05/08/2019
Flujo del sistema	El sistema debe informar que el usuario ha sido identificado correctamente. El sistema utilizará un modelo de reconocimiento que permita por medio de las imágenes previamente capturadas reconocer al usuario que ha ingresado al sistema.
Dependencias con:	
Revisado por:	**Realizado por:**
	JAIR HERNANDO VIDAL

Tabla 16. Requisito funcional 8 Cálculo de la distancia

Tipo:	Funcional
Módulo:	Cálculo
• **Cálculo de distancia**	
Descripción	La aplicación debe permitir calcular la distancia recorrida por el usuario.
Entradas	
Salidas	Notificar al usuario de la realización del proceso.
Fecha de captura:	05/08/2019

Flujo del sistema	El sistema deberá realizar el cálculo de la distancia tomando como insumo dos puntos con coordenadas (latitud, longitud) que corresponde a la ubicación inicial y final del usuario.
Dependencias con:	

Revisado por:	**Realizado por:**
	JAIR HERNANDO VIDAL

Tabla 17. Requisito funcional 9 Cálculo de la tarifa

Tipo:	Funcional
Módulo:	Cálculo
• **Cálculo de tarifa**	
Descripción	La aplicación debe permitir calcular la tarifa que el usuario debe cancelar.
Entradas	
Salidas	Notificar al usuario de la realización del proceso.
Fecha de captura:	05/08/2019
Flujo del sistema	El sistema deberá realizar el cálculo de la tarifa tomando como insumo la distancia recorrida y un valor por km recorrido.
Dependencias con:	

Revisado por:	**Realizado por:**
	JAIR HERNANDO VIDAL

Tabla 18. Requisito funcional 9 cálculo de la tarifa

Tipo:	Funcional
Módulo:	Notificaciones
• **Notificar la tarifa a pagar**	
Descripción	La aplicación debe notificar al usuario la tarifa que debe cancelar por el servicio prestado.
Entradas	
Salidas	
Fecha de captura:	05/08/2019
Flujo del sistema	▪ El sistema debe notificar al usuario por medio de un mensaje en pantalla el valor que debe cancelar el servicio prestado.
Dependencias con:	RF3
Revisado por:	**Realizado por:**
	JAIR HERNANDO VIDAL

Tabla 19. Requisito funcional 10 Almacenar datos

Tipo:	Funcional
Módulo:	Módulo
• **Almacenamiento temporal**	
Descripción	La aplicación debe permitir almacenar de manera temporal todas las imágenes de todos los usuarios que han ingresado al sistema.
Entradas	
Fuente	
Salidas	
Fecha de captura:	13/02/2020

Flujo del sistema	**Condiciones**: El almacenamiento de los archivos es de carácter temporal y solo mientras dure el viaje de cada uno de los ocupantes del automotor. Una vez finalice el proceso de identificación y el usuario haya cancelado el valor de la tarifa, el sistema debe eliminar los archivos correspondientes al usuario que ha finalizado su viaje.
Dependencias con:	

Revisado por:	**Realizado por:**
	JAIR HERNANDO VIDAL

7. DISEÑO

En este capítulo se describe el diseño de los diferentes componentes del sistema y sus interacciones. Se inicia con una descripción de cada uno de estos y se adiciona su imagen correspondiente. Además de describir las interfaces gráficas de usuario.

El sistema de identificación de pasajeros (Transass) es un software que permite controlar el pago de tarifas en transporte público intermunicipal utilizando técnicas de biometría como reconocimiento facial y huella dactilar. Este consta de 6 componentes: inicialización, captura de imagen, procesamiento de imagen, ubicación, persistencia, cálculo de tarifa y notificaciones.

7.1. Diagrama de componentes

A continuación, se describen cada uno de estos componentes y sus interacciones.

Componente de inicialización

Este permite iniciar el sistema, cargar las bibliotecas necesarias para su funcionamiento, establecer el espacio donde se almacenarán las imágenes que posteriormente se capturan y requerir del *componente de persistencia* para conectar una pequeña base de datos. No recibe ningún parámetro de ingreso y no retorna ningún valor.

Componente de captura de imágenes

Se encarga de realizar el proceso de captura de las imágenes del usuario. Inicialmente, captura la primera imagen del usuario al momento de ingresar al automotor y posteriormente, una segunda imagen antes de descender del vehículo con el fin de realizar el proceso de identificación. Por otra parte, se obtiene la ubicación que se tiene en el momento en que se captura cada una de las imágenes; para realizar lo anterior, se requiere del *componente de ubicación*, que nos proporciona los dos puntos geográficos que serán el insumo para realizar el cálculo de la distancia recorrida. También se hace uso del *componente de*

persistencia para guardar los puntos geográficos de inicio y fin, además, este componente detecta el rostro del usuario y validará la calidad de imagen antes de guardarla. Recibe como parámetro de ingreso un cuadro de imagen (frame) y retorna un archivo de imagen

Componente ubicación

Su objetivo es obtener la posición del usuario por medio de un módulo GPS; este retorna un punto de dos coordenadas que corresponde a la latitud y longitud del lugar geográfico. Recibe como parámetros de ingreso latitud y longitud y retorna un objeto que contiene las coordenadas (x, y).

Componente persistencia

Su objetivo es guardar algunos datos con el fin de ser utilizados posteriormente durante la permanencia del usuario en el sistema. Recibe como parámetros de ingreso: latitud, longitud, fecha de inicio y fin de viaje y retorna latitud, longitud, fecha de inicio.

Componente de procesamiento de imágenes

Su objetivo es realizar el proceso de identificación del usuario, tomando como base las imágenes capturadas por el *componente de captura*. Éste utiliza una técnica de reconocimiento facial para establecer si las dos imágenes corresponden al mismo usuario. Recibe como parámetros de ingreso una imagen en formato png y retorna un código y nombre de usuario y una variable de tipo Boolean (true/false); ésta última, indica si se logró identificar al pasajero.

Componentes de calcular tarifa

Su objetivo es calcular la tarifa que el usuario debe cancelar antes de abandonar el automotor en el destino. Para esto requiere que el *componente de procesamiento de imágenes* devuelva el código de usuario y un valor positivo que corresponda a la identificación del usuario, posteriormente obtener del *componente de persistencia,* el punto origen de recorrido y punto final, de esta

manera, realizar el cálculo de la distancia recorrida por el usuario, luego éste último valor, se multiplica por el costo asociado a la unidad de desplazamiento (kilómetro) dando como resultado el valor que debe cancelar el usuario como tarifa por el servicio prestado. Recibe como parámetros de ingreso variables de dos valores (true/false) si el pasajero fue identificado y código usuario; retorna la tarifa y código del usuario.

Componente de notificaciones

Encargado de notificar al usuario el valor de la tarifa que debe cancelar. Recibe como parámetros de ingreso el valor de la tarifa, el código del usuario y retorna el texto de mensaje para visualizar en pantalla.

Componente de huella dactilar

Se adiciona como una forma de identificación alternativa, su objetivo es gestionar las huellas dactilares de los usuarios que no pueden ser identificados en casos específicos. Cuando el componente de procesamiento de imágenes no es capaz de identificar a un usuario, se solicita al pasajero una huella dactilar al ingreso y al finalizar el viaje se captura una segunda huella con el fin de validar su identidad. Recibe como parámetros de ingreso un valor lógico (verdadero o falso) si el pasajero fue identificado y un código de usuario; retorna un valor lógico (verdadero / falso) si la huella es válida para un usuario.

Sintetizando, los componentes antes mencionados se visualizan en el siguiente diagrama de componentes (Véase Figura 26).

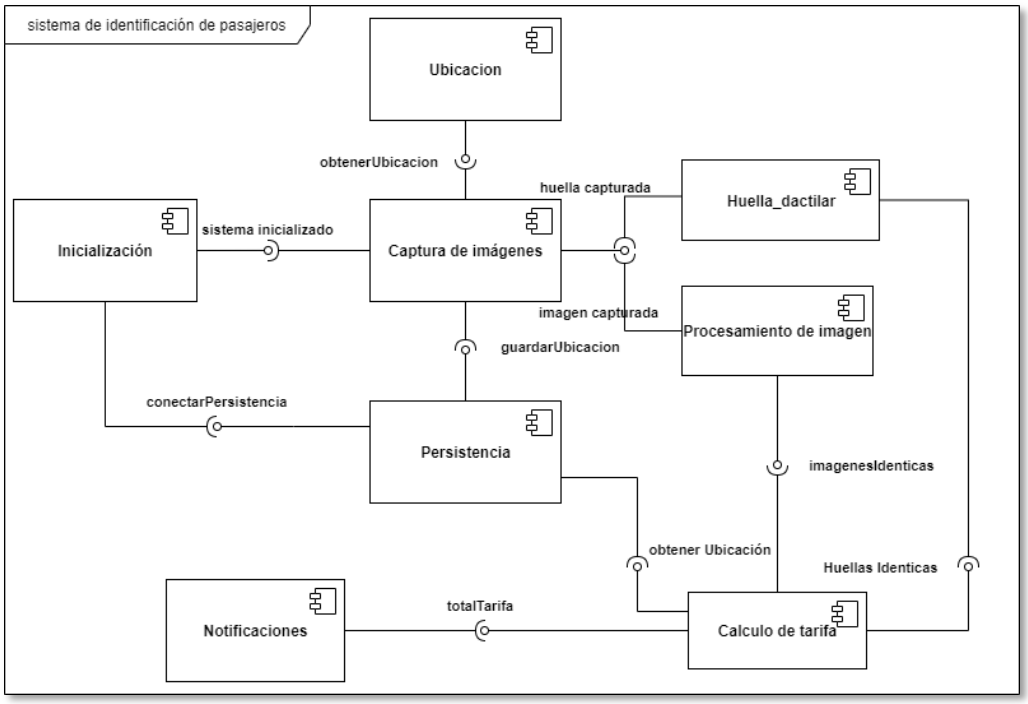


Figura 26. Diagrama de componentes.

7.2. Diagrama de secuencia

A continuación, se describe la Figura 27, es un diagrama de secuencia que describe el funcionamiento interno de la aplicación con mayor nivel de detalle.

- El sistema inicia cuando el *conductor* ingresa al sistema y permite inicializarlo por medio del método *enciendeVehiculo*(), que permite el suministro de potencia al dispositivo de hardware del sistema.
- Después de iniciado el hardware y con el sistema operativo en marcha, la aplicación inicia con el componente de inicialización que a su vez ejecuta los métodos *iniciarsistema*(), *crearCarpeta*() y *conectarPersitencia*().
- *iniciarsistema*() carga algunos archivos de configuración y bibliotecas.
- *crearCarpeta*() construye una carpeta donde se guardaran las fotos de los usuarios que ingresan al sistema.

- *conectarPersitencia*() permite la conexión a una pequeña base de datos local donde se guarda información necesaria del usuario para realizar cálculos posteriores.
- El método *activarCamara*(), permite detectar la cámara conectada al hardware e iniciar la transmisión de video.
- El *pasajero* ingresa al sistema y por medio del método *detectarUsuario*(), se detecta su rostro.
- Detectado el rostro del pasajero se procede a la captura de la primera imagen a través del método *capturarImagen*().
- Con método *terminarCaptura*() se verifica que previamente no haya sido guardada una imagen, si existe una significa que el usuario sale del sistema, de lo contrario el sistema interpreta que es un nuevo usuario y se continua con el flujo del programa.
- Con el método *validarImagen* () se revisa la calidad de la imagen, si la imagen es apta para realizar la identificación se guarda en el dispositivo, de lo contrario se procede con otros métodos de identificación.
- Cuando no se puede detectar al pasajero por su rostro, el sistema hace un llamado al método capturarHuella() encargado de obtener la huella dactilar del usuario como otro método de identificación.
- A través del método guardarPosicionHuella(), se guarda la huella dactilar del usuario y éste a su vez retorna la posición donde se guardó. Después de capturar la huella dactilar continua el flujo normal del programa.
- Ejecutando el método *capturarPosicion*(), se obtiene la ubicación geográfica del usuario y por medio del método *guardarGeo*(), se guarda en la base de datos la posición inicial del usuario;
- Por último, se procede a invocar el método *guardarDatosPasajero*(), que guarda la información restante del pasajero como: el código, la posición de la huella si la hay, el sitio o lugar de embarque, la fecha y hora de inicio de viaje.
- En segunda instancia, al momento de finalizar el viaje, el sistema verifica por medio del método *terminarCaptura()* que exista una imagen del usuario, si la

repuesta es positiva entonces se procede a capturar la segunda imagen, de lo contrario el sistema interpreta que es un nuevo usuario que ingresa al sistema por primera vez.

- Se procede a capturar la segunda imagen a través del método *capturaSegundaImagen*().
- Después el método *validarImagen*(), verifica la calidad de la segunda imagen y si la respuesta es positiva se guarda el archivo en el dispositivo, de lo contrario se procede con otros métodos de identificación.
- El método *buscarImagen*(), recorre el árbol de directorios que contiene la imagen de ingreso de cada uno de los usuarios y la retorna si es necesario.
- El sistema invoca el método *procesarCaptura*(), con el fin realizar el proceso de identificación. Este método utiliza reconocimiento facial con el fin de identificar al usuario comparando la imagen de ingreso con la imagen de salida, si el resultado de esta comparación es positivo el sistema ha logrado identificar al usuario, de lo contrario se procede a utilizar otros métodos para la identificación.
- Cuando no se puede identificar al pasajero por su rostro, el sistema hace un llamado al método capturarHuella() encargado de obtener la segunda huella dactilar del usuario como otro método de identificación.
- A través del método identificarHuella(), se busca la primera huella dactilar del usuario y se compara con la segunda huella capturada. Si las dos impresiones dactilares son idénticas se retorna la posición donde se guardó la primera huella. Después de capturar la huella dactilar continua el flujo normal del programa.
- Ejecutando nuevamente el método *capturarPosicion*(), se obtiene la segunda ubicación geográfica del usuario y por medio del método *guardarGeo*(), se guarda en la base de datos la posición final del usuario.
- Se invoca el método *guardarDatosPasajero*(), que actualiza la información restante del usuario: la posición de la segunda huella si la hay, el sitio o lugar de destino, la fecha y hora de fin de viaje.

- El sistema ejecuta el método *calcularDistancia*(), encargado de calcular la distancia recorrida por el usuario tomando como base un punto geográfico de inicio y un punto geográfico de fin utilizando una fórmula matemática se refuerza con una tabla de distancias obtenida del servicio de geolocalización de Google(Google Earth).
- *calcularTarifa*(), permite calcular el valor de la tarifa que el usuario cancelará; este método requiere como parámetro el resultado devuelto por *calcularDistancia*().
- El método *notificarUsuario*(), se encarga de informar al usuario el valor que debe pagar por el servicio prestado.
- El sistema no debe guardar información de ningún usuario, el método *eliminarInformacionUser*(), se encarga de realizar esta tarea, eliminando toda la información correspondiente a todos los usuario que han salido del sistema.

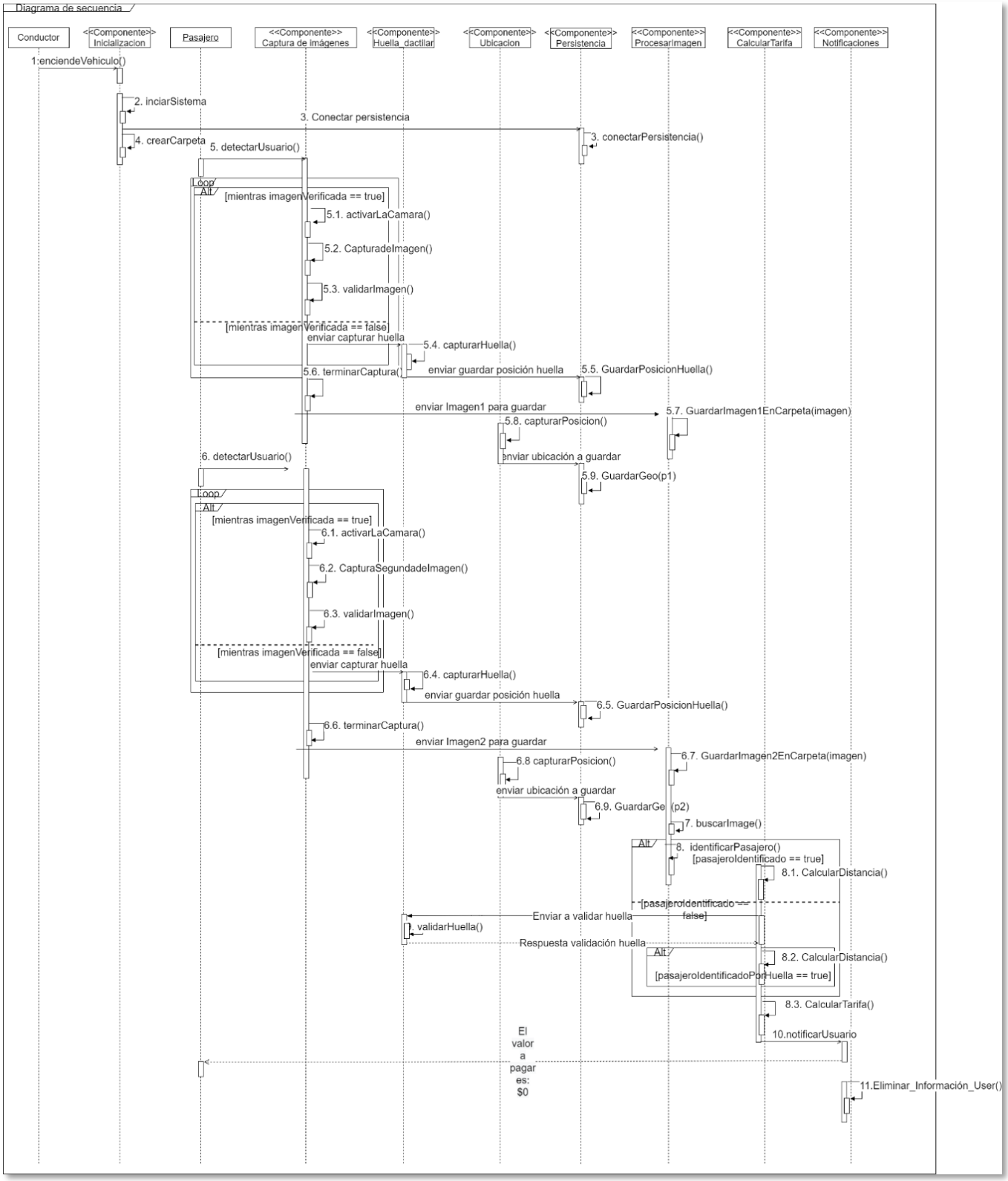


Figura 27. Diagrama de secuencia del sistema.

7.3. Diagrama de clases

La Figura 28 corresponde al diagrama de clases de la aplicación, el cual permite visualizar las colaboraciones entre componentes y que se describirá a continuación:

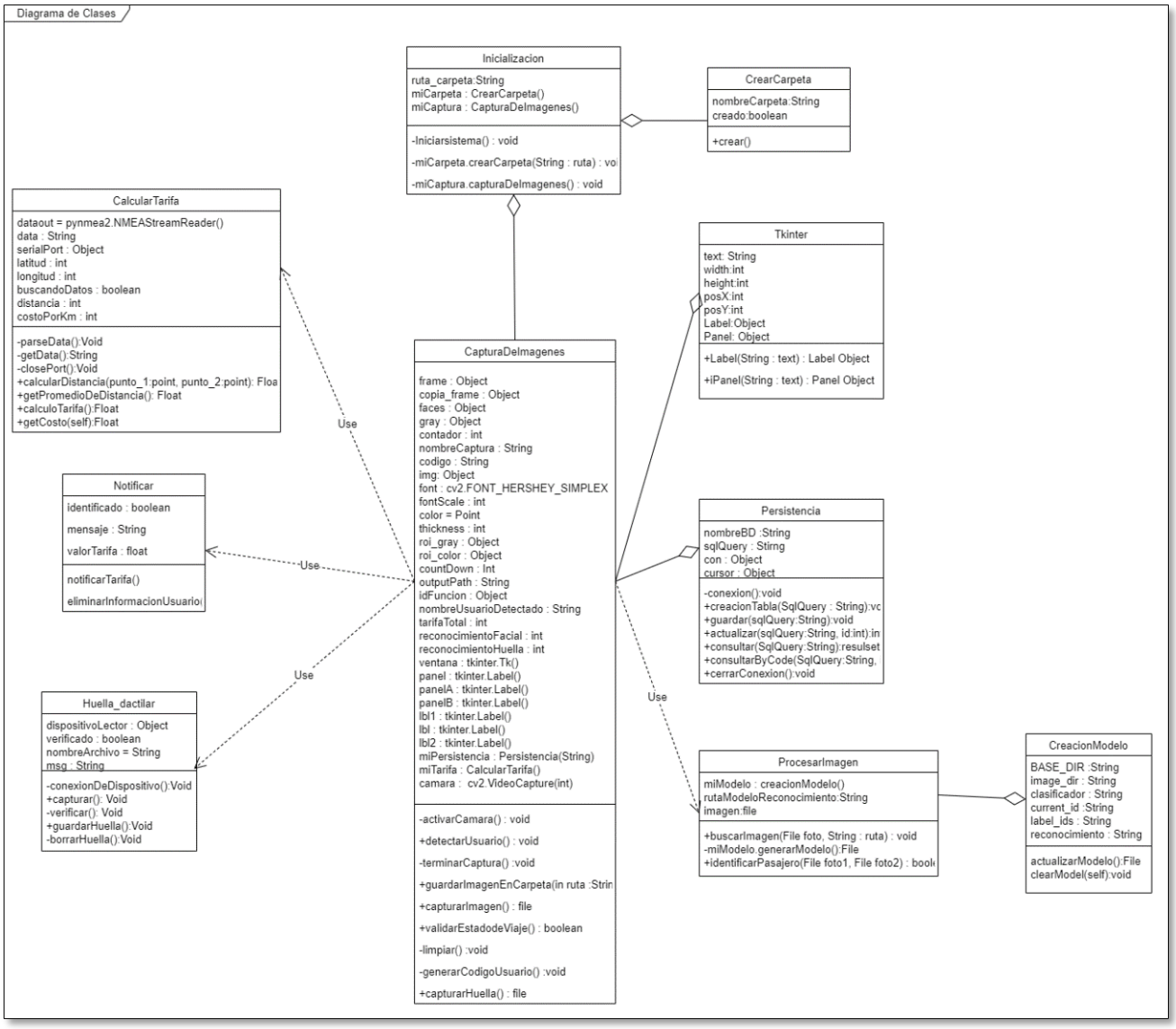


Figura 28. Diagrama de clases.

En este diagrama, la clase *Inicialización* se encarga de iniciar la aplicación, esta clase tiene como clases agregadas *crearCarpeta* y *CapturaDeImágenes*.

- La clase *crearCarpeta,* gestiona la creación de carpetas en el sistema, permite crear la carpeta donde se almacenará los archivos de cada usuario que ingresa al sistema.

- La clase *CapturaDeImágenes,* se visualiza como clase principal del sistema, ésta permite gestionar y controlar la transmisión de video e imágenes, ejecutar la interfaz gráfica de usuario y realiza otros procesos adicionales. Esta clase *usa* otras clases como *procesarImagen*, *CalcularTarifa*, *Notificar* y *Huellas*. Además, esta clase tiene como agregadas las clases *Tkinter* y *Persistencia*.

- La clase *Persistencia,* permite gestionar la pequeña base de datos que almacena datos de información geográfica.

- La clase *procesarImagen(),* permite gestionar el reconocimiento facial, además, esta clase tiene como agregada la clase *creacionModelo(),* que permite la creación del modelo de reconocimiento.

- La clase *creacionModelo(),* permite la creación del modelo de reconocimiento que utilizará la clase *procesarImagen(),* para realizar el reconocimiento facial.

- La clase *Tkinter* permite gestionar la interfaz gráfica de usuario, es una biblioteca para la creación de GUI que pertenece al lenguaje de programación de Python, se agrega a la clase *procesarImagen(),* encargada de gestionar la transmisión de video en el sistema.

- La clase *calcularTarifa(),* permite gestionar el cálculo de la tarifa en el sistema.

- La clase *notificar(),* permite gestionar los mensajes que emite el sistema y se encarga de eliminar los archivos cuando el usuario ha salido del sistema.

- La clase *huellas()* permite gestionar la captura de huellas dactilares en el sistema.

7.4. Diseño de interfaz de usuario

En la Figura 29, se visualiza la interfaz gráfica de usuario que tendrá la aplicación al iniciar. La interfaz en la parte superior posee el título de la aplicación, en el centro un marco donde se visualiza la transmisión de video de la cámara y en parte inferior una etiqueta que permite la visualización de mensajes o notificaciones del sistema.

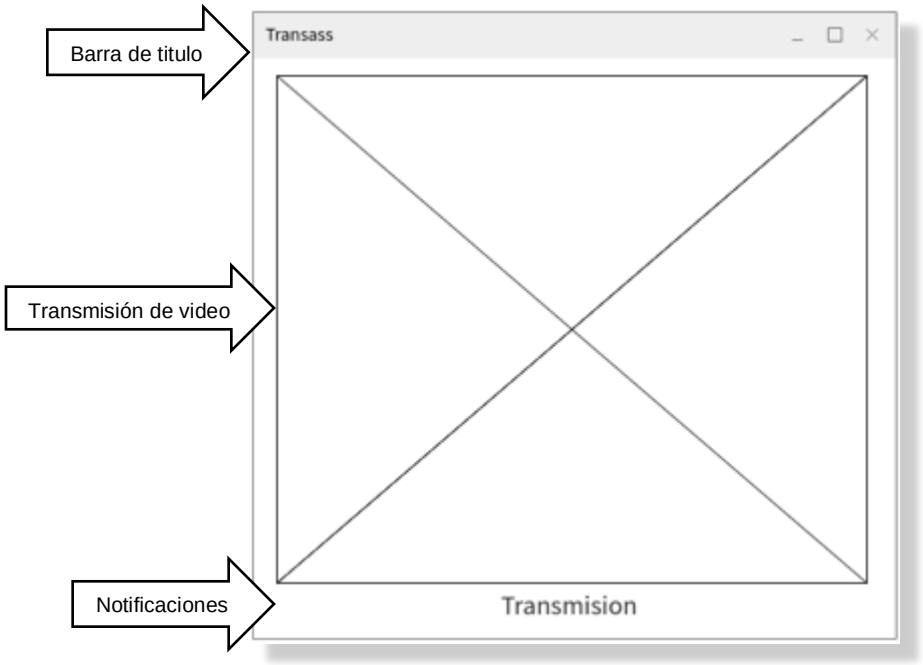


Figura 29. Interfaz gráfica inicial.

En la Figura 30, se visualiza la interfaz que tendrá la aplicación al momento de ingresar un usuario al sistema. La interfaz se divide en dos; en el lado izquierdo en su parte superior posee el título de la aplicación, en el centro un marco donde se visualiza la transmisión de video de la cámara, en la parte inferior una etiqueta que

permite la visualización de mensajes o notificaciones del sistema y, en el lado derecho, existe un marco donde se visualiza la captura de imagen que se hace al usuario, en este mismo sitio se realiza el proceso de identificación y en la parte inferior una etiqueta que permite la visualización de mensajes o notificaciones del sistema.

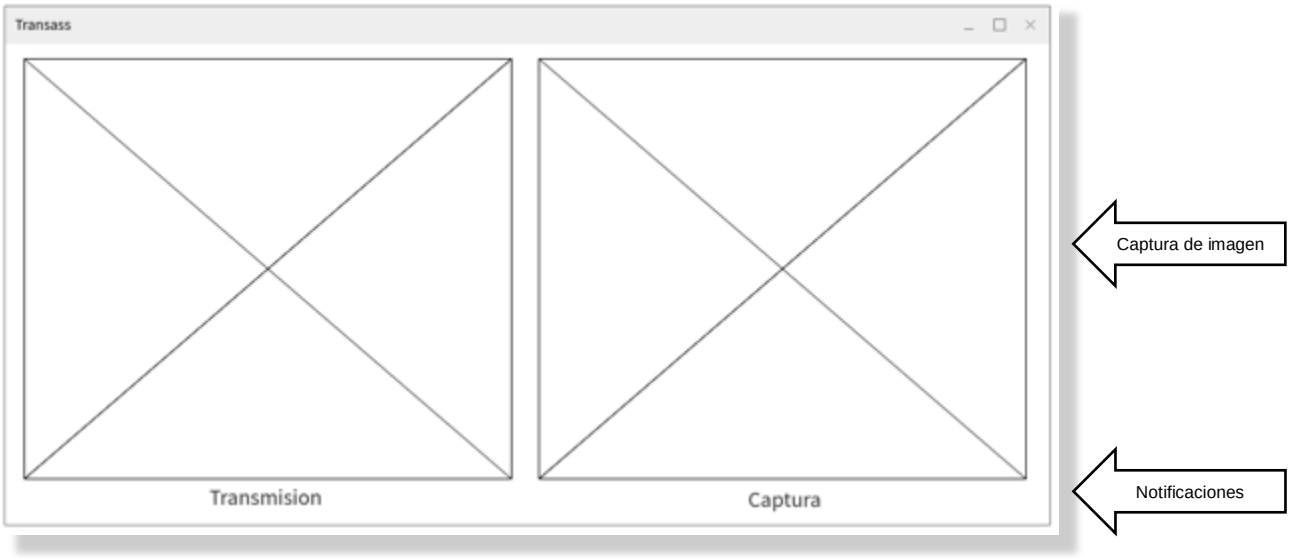


Figura 30. Interfaz gráfica ingreso de usuario.

7.5. Diagrama de datos

El sistema requiere una unidad de persistencia temporal, algunos datos son necesarios para los cálculos internos de distancia; por esta razón, utiliza una pequeña base datos con algunas tablas que guardan los datos de: rutas, distancias e información del usuario que ingresa al sistema, cabe recordar que estos datos solo se utilizan para realizar los cálculos de tarifa y posteriormente se eliminarán. En la Figura 31, se visualiza el diagrama de datos utilizado.

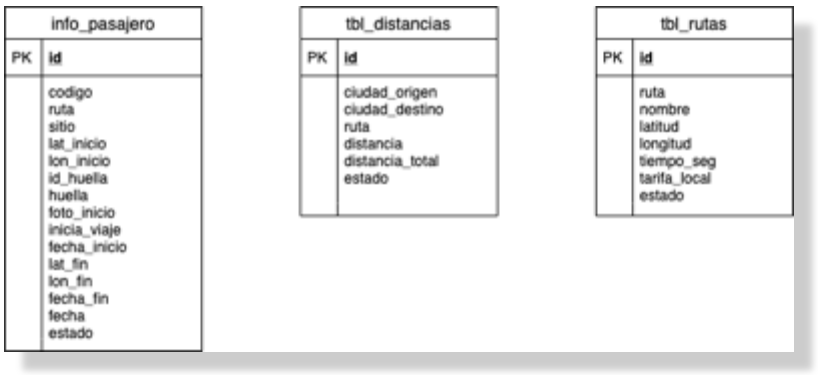


Figura 31. Tablas de datos utilizadas en el sistema.

7.6. Diagrama de despliegue

En la Figura 32 se visualiza un diagrama de despliegue que representa las interacciones entre los componentes de hardware y software.

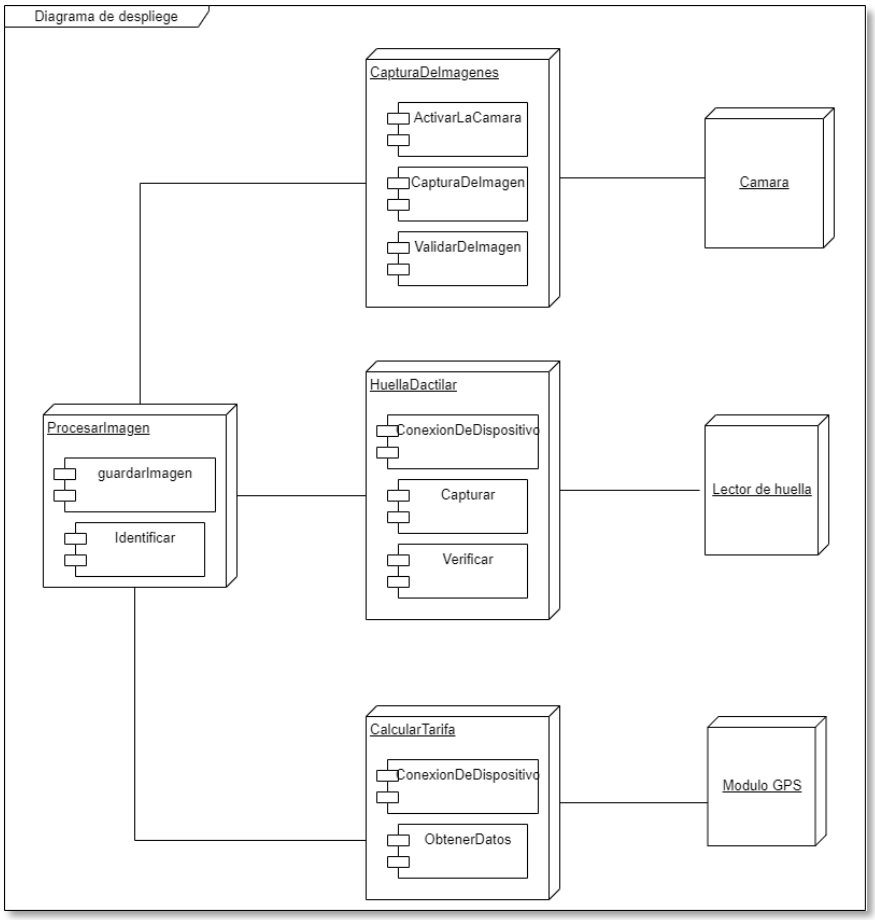


Figura 32. Diagrama de despliegue

El diagrama (Figura 32), permite visualizar la correspondencia existente entre las clases del sistema y los periféricos de hardware conectados al microcomputador. En este se visualiza la clase capturaDeImágenes(), que permite gestionar la cámara, que puede ser una webcam conectada por USB o conectada por un bus de datos. Posteriormente, la clase HuellaDactilar(), encargada de administrar el módulo lector de huella conectado por USB, y por último la clase CalcularTarifa(), que permite controlar el módulo GPS encargado de reportar la posición o ubicación del usuario.

8. IMPLEMENTACIÓN

En este capítulo se describe el proceso de implementación de la aplicación.

8.1. Descripción

La aplicación se inicia, por medio de un archivo ejecutable después de que el sistema operativo ha iniciado. En la Figura 33, se visualiza el escritorio del sistema operativo Raspbian y la aplicación ejecutándose con su interfaz inicial.

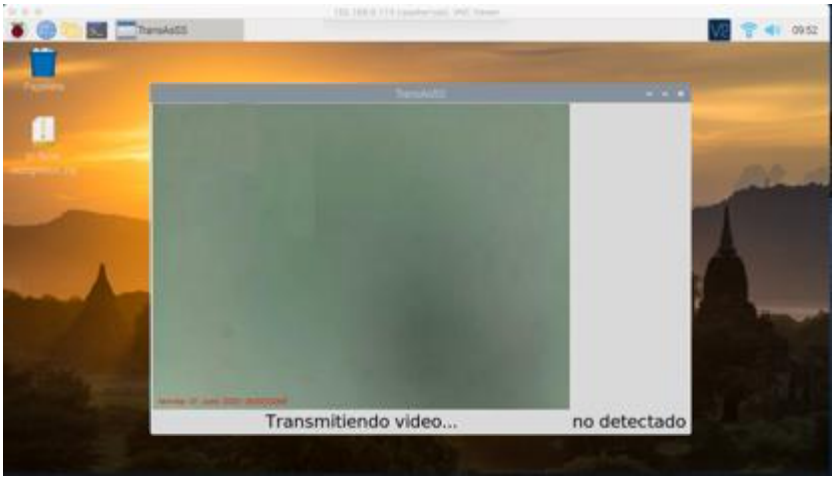


Figura 33. Escritorio de sistema operativo Raspbian.

Inmediatamente después de iniciar el sistema se inicia la transmisión de video, aunque la cámara posee una resolución de 720x1080, el marco (frame) de video es de 640 x 480, se limita el tamaño del marco con el fin optimizar el espacio disponible en la pantalla; con esta misma resolución se realiza las capturas de imagen.

En la parte superior de la ventana se visualiza la barra de título de la aplicación. El nombre que se ha puesto a la aplicación es TransAsS acrónimo de (Transport As

A Service o Transporte Como Servicio), puesto que la aplicación permite el cobro de tarifas por la distancia que ha recorrido del usuario.

En el lado izquierdo, existe un marco en el cual se visualiza la transmisión de video que proviene del módulo de cámara, en la parte inferior del marco se imprime la fecha y la hora, esto con el fin de informar al usuario el momento de ingreso al sistema. En caso de no existir el módulo de cámara conectado o una cámara web tipo USB conectada, el sistema no podrá iniciar. Hacia el lado derecho existe un espacio donde se carga la captura de imagen del usuario y se visualiza el proceso de detección e identificación.

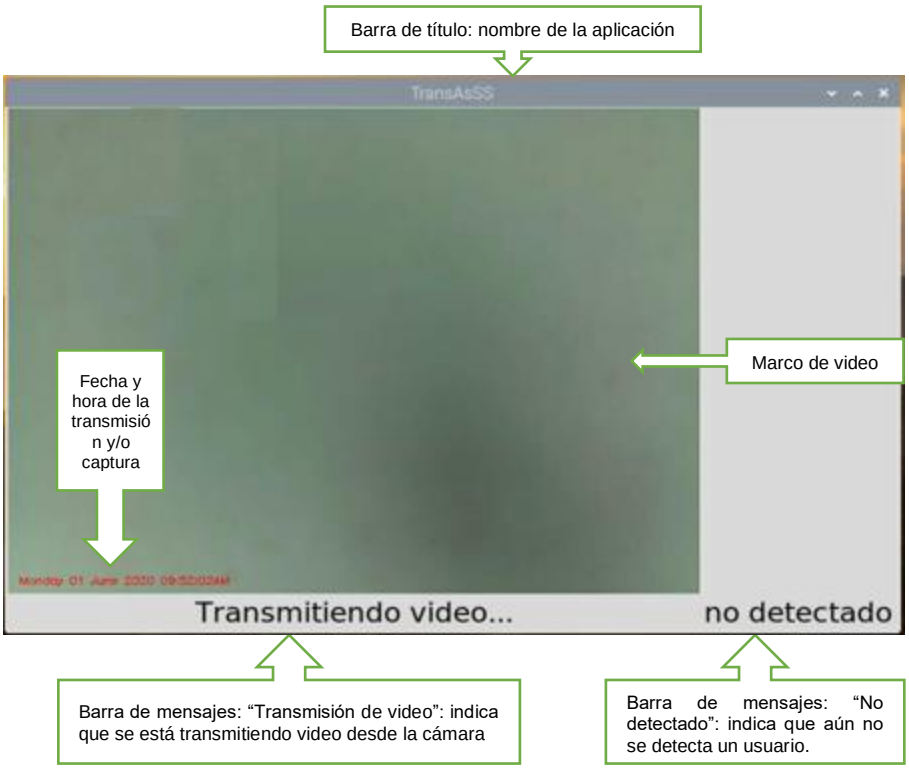


Figura 34. Interfaz de inicio del sistema.

Hacia la parte inferior de la ventana se encuentran dos etiquetas que permiten la visualización de mensajes o notificaciones: En el lado izquierdo, se visualiza información acerca del video; en el lado derecho, información acerca de la captura de la imagen.

8.2. Módulo detección y captura

La aplicación permite detectar a un usuario cuando se encuentra frente a la cámara, la detección se realiza usando el clasificador en cascada ya implementado en la biblioteca OpenCV. En la ventana del lado derecho de la interfaz se carga la captura de imagen (Véase Figura 35). El sistema dibuja un cuadro de color verde sobre el área del rostro del usuario y en la parte inferior se notifica “Detectado”.

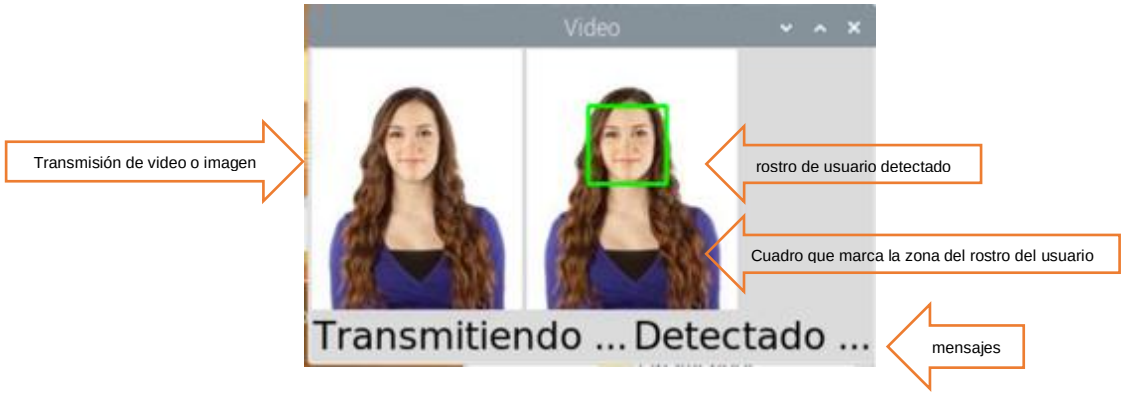


Figura 35. Detección de rostro.

Se requiere de un lapso de tiempo, entre 3 y 5 segundos de espera para poder realizar el proceso, debido a que el usuario podría no estar en posición o el lente desenfocado (véase Figura 36).

Figura 36. Captura de imagen.

Existe la posibilidad que el usuario se retire de la cámara, en cuyo caso, el sistema interrumpe la captura y si retorna de nuevo el proceso continuará.

```
1 faces = self.faceClassif.detectMultiScale(gray,1.3,5)
2 if len(faces) > 0:
3 for (x,y,w,h) in faces:
4 cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
5       rostro = auxFrame[y:y+h,x:x+w]
6       self.lbl2['text'] = "Detectado ..."
7             #se visualiza la captura
8              fr1 = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
9              self.imagenActual = Image.fromarray(fr1)
10               fr1 = ImageTk.PhotoImage(image=self.imagenActual)
11               self.panelB.config(image=fr1)
12               self.panelB.image = fr1
13               self.lbl2['text'] = "Capturando ..."
14 p = os.path.sep.join([self.savePath+str("/user_"+self.código),"{}.png".format(
 str(self.código).zfill(0)) ] )
15 cv2.imwrite(p,  rostro )
```

La detección del rostro se realiza por medio de la clase detectarImágenes(), como primera medida se carga el archivo *Haarcascade_frontalface_alt.xml, un* clasificador en cascada Haar que permite detectar rostros en posición frontal y utilizando la función detectMultiScale() (Línea 1) con los siguientes parámetros:
gray = captura de imagen en escala de grises.

scaleFactor = 1.3: intenta buscar rostros escalando la imagen varias veces. Este factor indica que tanto se puede reducir la imagen. El valor 1.3 indica que se reducirá la imagen en un 30%. Si el valor es muy grande se pierden detecciones, puesto que sólo toma en cuenta las más importantes, en caso contrario tomará un poco más de tiempo, debido a que se incrementan las detecciones [50].

minNeighbours = 5: indica la calidad de las detecciones: un valor muy alto resulta en menos detecciones, pero con más fiabilidad [50].

MinSize = (30,30):indica el tamaño que tendrá cada uno de los rostros detectados, rostros por debajo de este valor no se tendrán en cuenta [50].

Después de la detección se procede a capturar la imagen. La función cv2.imwrite permite crear un archivo de imagen a partir de un marco o frame de video o imagen cargado. De esta manera, construye un archivo de imagen en formato *.png, y se adiciona a una subcarpeta construida previamente con el siguiente nombre user_27ec34ff, donde “user_” es una cadena de texto y “27ec34ff” es un código generado por la función Random del lenguaje de programación Python y se guarda en la unidad de almacenamiento del hardware. En la Figura 38 se visualiza la subcarpeta creada en el sistema y en la Figura 37 el archivo de imagen.

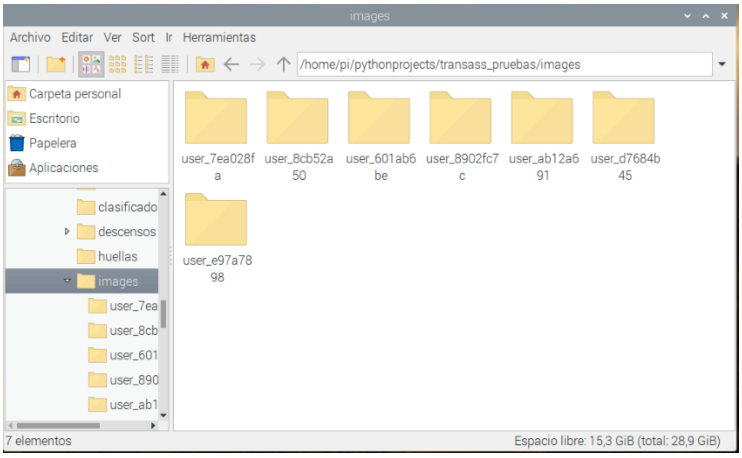


Figura 38. Subcarpetas de usuarios ingresados.

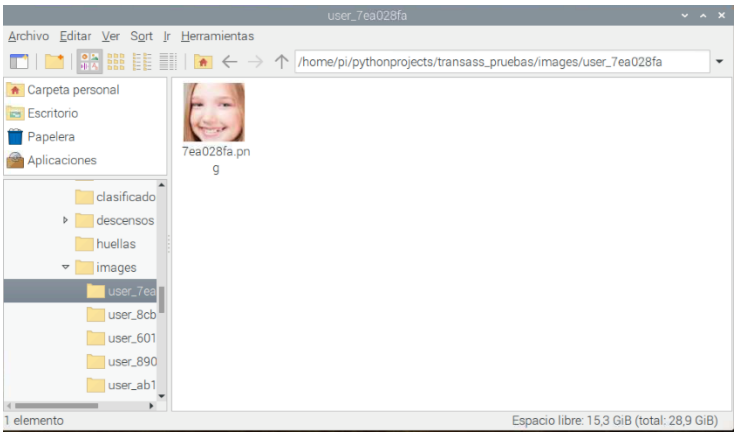
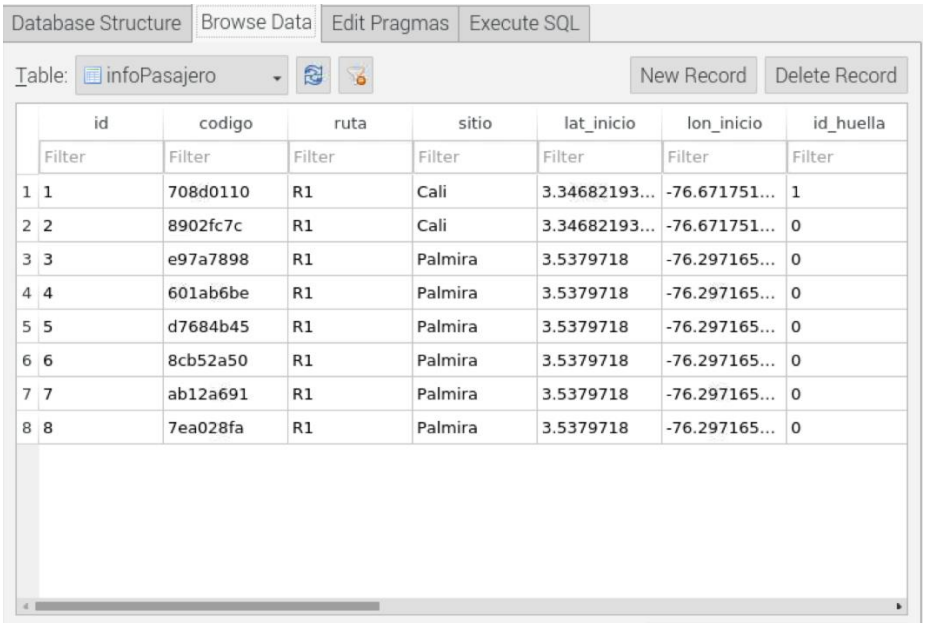


Figura 37. Imagen capturada.

Hasta este momento, se ha capturado la primera imagen que indica que el usuario ha entrado al sistema. La ubicación inicial (véase Figura 39) del usuario se guarda en una pequeña base de datos. Este proceso se realiza después de guardar la imagen y es gestionado por la clase Persistencia.py. esta clase contiene todas las funciones básicas que permite gestionar una base datos. Algunos métodos importantes son conexión(), guardar(), actualizar() y cerrarConexion().



	id	codigo	ruta	sitio	lat_inicio	lon_inicio	id_huella
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	708d0110	R1	Cali	3.34682193...	-76.671751...	1
2	2	8902fc7c	R1	Cali	3.34682193...	-76.671751...	0
3	3	e97a7898	R1	Palmira	3.5379718	-76.297165...	0
4	4	601ab6be	R1	Palmira	3.5379718	-76.297165...	0
5	5	d7684b45	R1	Palmira	3.5379718	-76.297165...	0
6	6	8cb52a50	R1	Palmira	3.5379718	-76.297165...	0
7	7	ab12a691	R1	Palmira	3.5379718	-76.297165...	0
8	8	7ea028fa	R1	Palmira	3.5379718	-76.297165...	0

Figura 39. Registros de ingreso al sistema.

8.3. Módulo de reconocimiento

Este componente tiene dos funciones principales; la primera, entrenar al reconocedor de rostros y; la segunda, realizar la identificación o reconocimiento de los usuarios.

8.3.1. Entrenamiento del clasificador o reconocedor

El entrenamiento es gestionado por la clase creacionModelo.py y el método actualizarmodelo(), encargado de realizar el entrenamiento. Inicia definiendo algunas variables que se requieren para realizar el proceso. El arreglo Labels

(línea 3), guarda las etiquetas que se asignan a cada imagen de cada usuario, el arreglo facesData(línea 4), guarda cada una de las imágenes que ha capturado el sistema, luego la variable label(línea 5), asigna una etiqueta distinta para cada captura. Es decir, se necesita informarle al computador quienes son las personas que se van a identificar y por eso se requiere que todas tengan “etiquetas” diferentes (Véase Figura 40).

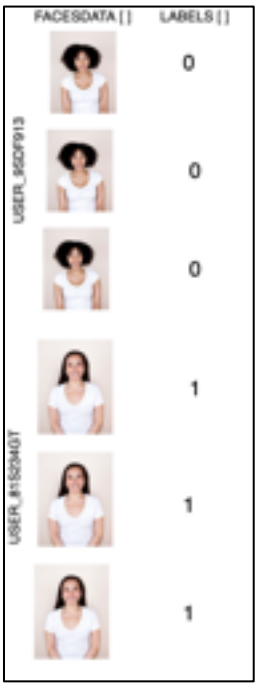


Figura 40. Asignación de etiquetas a las imágenes.

Se inicia el recorrido de toda la estructura del directorio, donde se guardan las imágenes (línea 6), se lee el contenido de cada carpeta (línea 7), adicionando el valor de la variable label a la lista labels, (línea 10); acto seguido, se añade al arreglo facesData la imagen encontrada en la carpeta actual (línea 11). Se carga la imagen y se visualiza a través de la interfaz. (líneas 12 y 13).

Se crea un objeto de tipo reconocedor (línea 19), posteriormente con la función train() del objeto (línea 20), se procede a entrenar al clasificador. Para realizar este proceso, se pasa como parámetros la lista de etiquetas y la lista imágenes

previamente construidas. Finalmente, para ahorrar tiempo en el entrenamiento (línea 22) la función write(), permite guardar el entrenamiento realizado con anterioridad, de esta manera se escribe en un archivo *.xml. el modelo que nos servirá para reconocer rostros y que se utilizará en el reconocimiento facial.

```
1 def actualizarModelo(self):
2     peopleList = os.listdir(self.dataPath)
3     labels = []
4     facesData = []
5     label = 0
6     for nameDir in peopleList:
8         personPath = self.dataPath + '/' + nameDir
9         for fileName in os.listdir(personPath):
10            labels.append(label)
11            facesData.append(cv2.imread(personPath+'/'+fileName, 0))
12            image = cv2.imread(personPath+'/'+fileName,0)
13            cv2.imshow('image',image)
14            cv2.waitKey(10)
15            time.sleep(0.5)
16        label = label + 1
17        time.sleep(0.5)
19    face_recognizer = cv2.face.LBPHFaceRecognizer_create()
21    face_recognizer.train(facesData, np.array(labels))
      22        face_recognizer.write('modeloLBPHFace_ing_1.xml')
```

8.3.2. Reconocimiento facial

El reconocimiento facial es gestionado por la clase procesarImagen().py, que contiene el método identificar(), encargado de realizar el reconocimiento facial, este usa un modelo de entrenamiento (línea 5), previamente creado por la clase creacionModelo.py. inicia con la detección de rostros (línea 6) con el fin de permitir imágenes que contengan rostros. Cargando la imagen capturada del usuario usando la función cv2.imread() se adiciona a la variable frame (línea 7), convirtiendo ésta a escala de grises (línea 8). Se obtiene una copia de la imagen cargada (línea 9) asignándola a la variable auxframe. Se aplica el clasificador con el fin de detectar el rostro que existen en la imagen (línea 10), se evalúa la cantidad de

rostros detectados (línea 11), luego de la variable auxFrame (línea 13), se recorta solo el área del rostro y se redimensiona la imagen a 150x150 pixeles (línea 14), con el objetivo de que todas las caras sean del mismo tamaño, después con la función predict(rostro) se trata de predecir a quien corresponde el frame o marco de imagen que se envía como parámetro (línea 15).

```
1 def identificar(self):
2     try:
3         contador = 0;
4         face_recognizer = cv2.face.LBPHFaceRecognizer_create()
5        face_recognizer.read('modeloLBPHFace_ing_1.xml')
6        faceClassif= cv2.CascadeClassifier(cv2.data.Haarcascades+'Haarcascade_frontalface_default.xml')
7         frame = cv2.imread(self.imagePath)
8         gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
9         auxFrame = gray.copy()
10          faces = faceClassif.detectMultiScale(gray,1.3,5)
11          if len(faces) > 0:
12              for (x,y,w,h) in faces:
13                  rostro = auxFrame[y:y+h,x:x+w]
14                  rostro = cv2.resize(rostro,(150,150),interpolation= cv2.INTER_CUBIC)
15                  resul = face_recognizer.predict(rostro)
16                  if resul[1] < 80:
17                      contador += 1
18                      return self.imagePaths[resul[0]], resul[1]
19                  else:
20                      return None, 0
21          else:
22              return None, 0
23      except Exception as e:
24          print(" ERROR: " + str(e))
```

La variable resul (línea 16), guarda el resultado que devuelve la función predict(), después de aplicar el reconocimiento. La predicción se realiza con base al modelo de entrenamiento creado con anterioridad, si el frame o marco de imagen se parece o no a alguno de los frames o marcos de imagen utilizados para el entrenamiento. La variable resul es un arreglo de dos posiciones y contiene los siguientes valores:

- id: es la posición de la etiqueta que le corresponde al usuario en lista utilizada en el entrenamiento.
- confianza: Es un valor que indica que tan parecidas o cercanas (distancia) son ambas imágenes.

Los valores de confianza nos sirven para evaluar la calidad del reconocimiento y con base a esta medida, se toman decisiones basándose en las siguientes condiciones:

Si confianza tiende a cero: nos indica que la imagen que se desea identificar es muy parecida a las imágenes utilizadas durante el entrenamiento[50].

Si la confianza es mayor de 90: nos indica que la imagen que se desea identificar es muy poco parecida a las imágenes utilizadas durante el entrenamiento.
Cabe resaltar que el valor de confianza puede variar por condiciones de iluminación, posición del rostros, enfoque del lente o pocas imágenes utilizadas durante el entrenamiento[50].

En algunos casos el reconocedor de OpenCV puede presentar fallas en su desempeño debido a: la posición del rostro, vibración, etcétera, hace que la función de detección se debilite. Como una opción que permita mitigar el impacto en el sistema se adiciona un lector de huella dactilar, se agregan funciones al software con el fin de gestionar las huellas. Al iniciar el sistema, lee un archivo de parámetros que le indican que modo debe iniciar si solo reconocimiento facial o adicionar huella dactilar para reforzar la identificación. (Véase Figura 41).

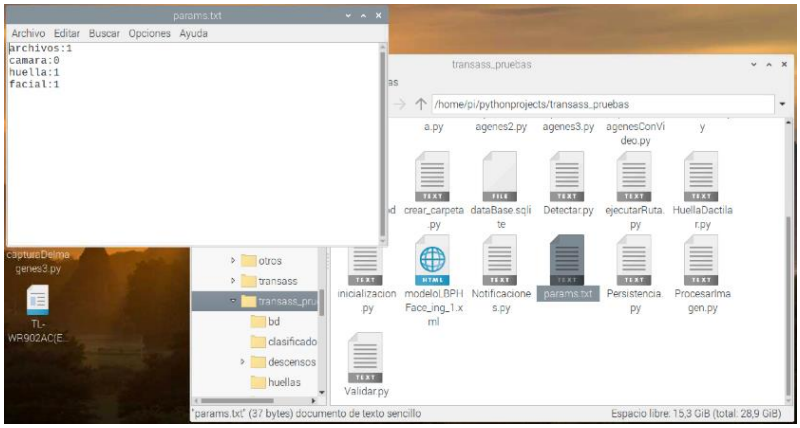


Figura 41. Ubicación y contenido del archivo params.txt.

La captura de la huella se hace después de validar si la foto capturada al ingresar no contiene un rostro o la posición del rostro no es frontal o no se reconoce el rostro a causa de algún obstáculo. El sistema envía el siguiente mensaje:

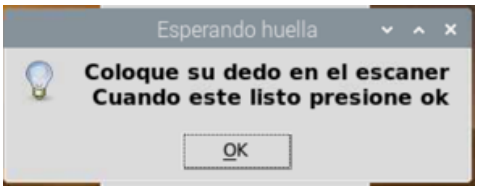


Figura 42. Mensaje para captura de huella.

Espera hasta que el usuario coloque alguno de los dedos de su mano en el lector de huella conectado al hardware. Cuando el usuario termina el viaje la aplicación vuelve a solicitar la huella dactilar con el fin de identificarlo.

La clase HuellaDactilar.py encargada de gestionar las huellas, contiene el método capturarHuella(self). La captura de huella se realiza esperando a que usuario coloque su dedo en el escáner (línea 8) luego la captura se convierte al formato que requiere el dispositivo (línea 10) luego se valida si ya existe una huella idéntica(línea 11), se evalúa si existe o no (línea 13) solo en el caso que no se repita la huella (línea 17) se devuelve la posición, que corresponde a la ubicación donde se guardó la impresión.

```
1   def capturarHuella(self):
2   try:
3   print('Esperando capturar una huella...')
4   ventana = tk.Tk()
5   ventana.title("consulta")
6   messagebox.showinfo("Esperando huella","Coloque su dedo\n presione ok",  parent=ventana)
7   ventana.destroy()
8   while ( self.dispositivoLectorDeHuellas.readImage() == False ):
9   time.sleep(0.5)
10  self. dispositivoLectorDeHuellas.convertImage(0x01)
11  result = self. dispositivoLectorDeHuellas.searchTemplate()
12  positionNumber = result[0]
13  if ( positionNumber > 0):
```

```
14  print('La impresion dactilar ya existe#' + str(positionNumber))
15  return 0
16  else:
17  positionNumber = self. dispositivoLectorDeHuellas.storeTemplate()
18  print('Nueva impresion dactilar en la posicion #' + str(positionNumber))
19  return positionNumber
20  except Exception as e:
21  print('Exception message: ' + str(e))
22  exit(1)
```

El método verificarhuella(), permite validar si la huella ingresada previamente corresponde a la segunda huella del usuario; para realizar este proceso, se comparan las características (línea 13) de las dos impresiones capturadas y se retorna la posición(línea 15) en caso contrario no se retorna ningún valor.

```
1 def verificar(self):
2 try:
3 print('Esperando nuevamente capturar una huella...')
4 ventana = tk.Tk()
5 ventana.title("consulta")
6 messagebox.showinfo("Esperando huella","Coloque su dedo\n presione ok",  parent=ventana)
7 ventana.destroy()
8 while ( self.dispositivoLectorDeHuellas.readImage() == False ):
9 pass
10 self.dispositivoLectorDeHuellas.convertImage(0x01)
11 result = self.dispositivoLectorDeHuellas.searchTemplate()
12 positionNumber = result[0]
13 if ( self.dispositivoLectorDeHuellas.compareCharacteristics() == 0 ):
14 print("La huella NO es igual")
15 return positionNumber
16 else:
17 print("La huella ES igual")
18 return 0
19 except Exception as e:
20 print('Mensaje error:' + str(e))
```

Cuando el usuario desciende del vehículo, se realiza la siguiente comprobación:

```
if códigoUsuarioObtenidoBD == códigoUsuarioIdent[1]
and posicionHuellaSecundaria == int(posicionHuellaInicialObtenidaBD):
```

Se compara el código del usuario identificado o reconocido por el sistema y el código inicial guardado en la bd, además, se compara la posición de la huella actual con la posición de la huella guardada en la bd, si el resultado es verdadero, se calcula la tarifa y se notifica; en caso contrario, se requiere que el usuario ingrese manualmente datos de ubicación y origen de viaje para notificar la tarifa.

8.4. Módulo cálculo y notificación.

La clase calcularTarifa.py encargada de calcular la tarifa del usuario involucra varios métodos; inicialmente, se requiere obtener la ubicación final, y para ello, el método getData(), permite obtener la posición de latitud y longitud que proviene del módulo GPS conectado al hardware. Este método hace uso de la función parseData(), que permite la conversión a texto de la información que proviene del GPS, además, es el encargado de retornar las posiciones para crear el punto o ubicación final del recorrido.

```
1def parseData(self):
2       if self.data[0:6] == "$GPGGA":
3          newmsg=pynmea2.parse(self.data)
4          self.latitud = newmsg.latitude
5          self.longitud = newmsg.longitude
6          self.buscandoDatos = False
7          return self.latitud, self.longitud
```

```
1 def getData(self):
2       self.serialPort = serial.Serial('/dev/ttyUSB0', 9600,
3                   parity=serial.PARITY_NONE, stopbits=serial.STOPBITS_ONE,
4                   bytesize=serial.EIGHTBITS, timeout=1)
5       while self.buscandoDatos:
6         self.data = self.serialPort.readline().decode('ascii', errors='replace')
7         time.sleep(0.01)
8         if self.data:
9            self.parseData()
```

Para el cálculo de la distancia se implementó la fórmula del semiverso[21][20] en el método getDistancia().

```
1 def getDistancia(self, lat1, lon1, lat2, lon2):
2       r = 6371000
3       c = pi/180
4 d = 2 * r * asin( sqrt( sin( c* (lat2-lat1) / 2)**2 + cos(c*lat1) * cos(c*lat2) *sin(   c * (lon2-lon1) / 2)**2 ) )
5     dist = round( (d/1000), 0)
6     return dist
```

El procedimiento para obtener la distancia, requiere de un punto inicial que se guarda en una pequeña base de datos después de capturar la primera imagen, y que sirve de consulta para realizar el cálculo de la distancia. El punto dos se obtiene después de capturar la segunda imagen; obteniendo las ubicaciones, estas, se trasladan como parámetros a la función getDistancia(lat1, lon1, lat2, lon2) (línea 1). Se usa el radio de la tierra en metros (línea 2), calculando el ángulo en radianes (línea 3), aplicando la fórmula (línea 4), el resultado se divide entre 1000 con el fin realizar la conversión a kilómetros y se retorna el valor (líneas 5 y 6).

Hasta este momento se tiene el valor de la distancia recorrida, sin embargo, este valor debe ser validado puesto que la fórmula del semiverso es una estimación de cálculo y podría existir un margen de error con una distancia real terrestre. Aunque existen en el mercado dispositivos GPS que son capaces de calcular la distancia recorrida con una excelente precisión, son muy costosos; además, el módulo GPS utilizado solo tiene funciones básicas y no es capaz de realizar tal cálculo.

Tabla 20. Tabla de distancias.

Ciudad origen	Ciudad destino	Ruta	Distancia
Cali	Palmira	R1	33
Palmira	Cali	R2	473
Palmira	Cali	R1	33
Cali	Jamundí	R3	23
Jamundí	Cali	R4	149
Palmira	Guadalajara de Buga	R1	80
Guadalajara de Buga	Palmira	R1	80
Guadalajara de Buga	Palmira	R2	440
Jamundí	Villa Rica	R3	38
Villa Rica	Jamundí	R4	126
Guadalajara de Buga	Tuluá	R1	105
Tuluá	Guadalajara de Buga	R1	105
Tuluá	Guadalajara de Buga	R2	393
Villa Rica	Santander de Quilichao	R3	53
Santander de Quilichao	Villa Rica	R4	111

Tuluá	Bugalagrande	R1	123
Bugalagrande	Tuluá	R1	123
Bugalagrande	Tuluá	R2	368
Santander de Quilichao	Mondomo	R3	68
Mondomo	Santander de Quilichao	R4	96
Bugalagrande	La Paila	R1	133
La Paila	Bugalagrande	R1	133
La Paila	Bugalagrande	R2	350
Mondomo	Pescador	R3	88
Pescador	Mondomo	R4	81
La Paila	La Tebaida	R1	174
La Tebaida	La Paila	R1	174
La Tebaida	La Paila	R2	340
Pescador	Tunía	R3	113
Tunía	Pescador	R4	61

Con base a lo anterior, el valor de la distancia se valida comparando el resultado obtenido con una tabla de valores que han sido previamente calculados y certificados(Véase Tabla 20).

El cálculo final de la tarifa puede variar dependiendo de la posición obtenida por el módulo GPS.

- Si el usuario no ha salido del perímetro urbano se aplicará una tarifa local.
- Si el usuario ha salido del perímetro urbano se calcula la tarifa con base a la distancia calculada, se compara con la tabla de distancias y la ruta asignada.

Por último, se notifica el valor de la tarifa, enviando una cadena de texto hacía la interfaz donde se visualizará. Véase Figura 43.

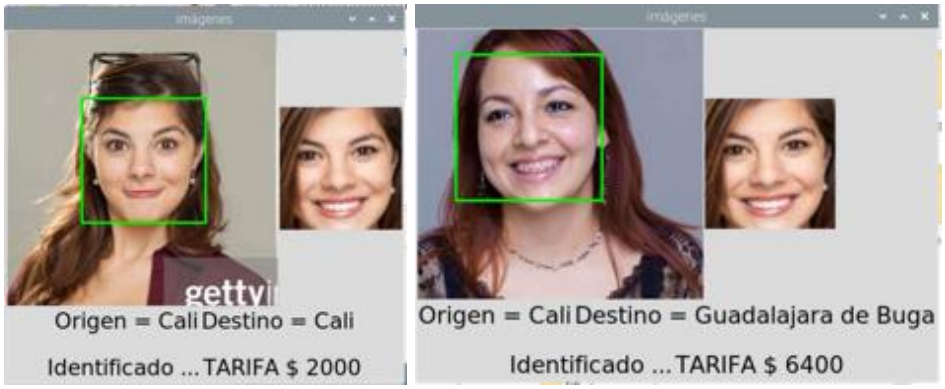


Figura 43. Reconocimiento del usuario y notificar el valor de la tarifa.

8.5. Dificultades durante la implementación

- Se inició la implementación con un microcomputador Raspberry pi versión B, pero el desempeño de este no era muy bueno para la cámara, haciendo que esta fuera más lenta para capturar o transmitir el video, por esta razón se consigue un hardware más robusto pasando a un Raspberry pi 4, que posee mejores prestaciones, perfeccionando sustancialmente aspectos como la transmisión de video, la captura de la imagen y el retardo en el enfoque.
- En el desarrollo de este prototipo, la fuente de video es vital, por esta razón, se utilizaron dos cámaras con diferente interfaz de conexión. Aunque las dos tienen muy buenas prestaciones, surgieron inconvenientes al momento de capturar imágenes con poca luz, es el caso de la cámara especificada en la Tabla 6, que aunque posee un sistema visión nocturna se notaban problemas al momento de capturar o reconocer las imágenes, algunos como: si en el ambiente la luz es tenue o semi oscuro(Véase Figura 44) o si hay luz detrás del usuario.

Figura 44. Captura de imagen con luz tenue o poca.

- Detectar la ubicación del usuario en el inicio y al final del recorrido siempre fue un reto en el desarrollo del proyecto. Con el uso de un módulo GPS (Véase

Tabla 8) que se integra al hardware, se pueden obtener estos valores y de esta manera calcular la distancia recorrida.

- En el desarrollo del proyecto, se planteó el uso de algún lenguaje de programación orientado a objetos, aunque en el inicio se consideró a JAVA, como el lenguaje elegido, múltiples razones hicieron desistir de la elección, algunas como: la obsolescencia de algunas bibliotecas para el procesamiento de imágenes, la ralentización en el procesamiento de los datos y problemas en la comunicación entre el entorno de desarrollo y el microcomputador, entre otras. Así que nació la necesidad de buscar otras opciones, hasta tomar la decisión de utilizar a Python como el lenguaje de programación para el desarrollo, con algunas razones de peso como: la madurez en el procesamiento de imágenes, la actualización constante de bibliotecas o paquetes de desarrollo (SDK), simplicidad en el código y la flexibilidad en la sintaxis; además de su rápida compilación y ejecución.

8.6. Ajustes posteriores a la aplicación de las pruebas

Después de ejecutar el plan de pruebas se requirió ajustar nuevamente el software, en la siguiente sección se describen los cambios que se aplicaron:

- **El usuario no es detectado al ingresar al sistema**: En ciertos casos, cuando el sistema no puede detectar el rostro del usuario por una de las siguientes razones: la captura de imagen ingresada presenta problemas con la posición del rostro, no contiene un rostro, la calidad de imagen es deficiente, el lector de huella no captura correctamente o la impresión dactilar no es de buena calidad, etcétera. El sistema asigna una imagen genérica a cada usuario que no se ha detectado, con el fin de permitirle el ingreso y al finalizar el viaje la aplicación solicita identificarse de forma manual solicitando información externa que debe ser suministrada por el usuario. Por esta razón, se carga una interfaz con imágenes genéricas que además visualizan un código aleatorio asignado automáticamente, la

ubicación donde ingresó al vehículo, la fecha y hora (véase Figura 45) y una caja de texto que permite el ingreso del código (véase Figura 46). Esta es otra forma de identificar al usuario en caso extremo de fallar los dos métodos expuestos anteriormente.

Figura 45. Listado de usuarios sin imagen inicial.

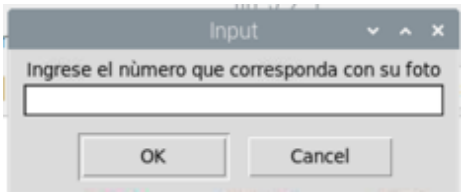


Figura 46. Caja de texto para ingreso de código de usuarios no detectados.

- **El usuario no es identificado o reconocido por el sistema al finalizar el viaje**: en algunos casos al finalizar el viaje el sistema no puede identificar o reconocer al usuario, ya sea porque la captura de imagen ingresada presenta problemas como: falta de iluminación, posición del rostro o no contiene un rostro, etcétera. Para dar solución a este problema, se requiere identificarse de forma manual, solicitando información externa que debe ser suministrada por el usuario, por esta razón, se carga una interfaz con los rostros de las personas, un código aleatorio asignado automáticamente, la fecha y hora de ingreso (véase Figura 47) y una caja de texto que permite el

ingreso del valor (véase Figura 48). Esto permite indicarle al aplicativo identificar al usuario desconocido y notificar el valor de la tarifa.

Figura 47. Lista de imágenes de usuarios.

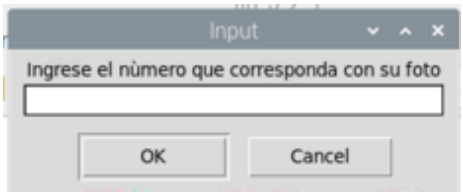


Figura 48. Caja de texto para ingreso de código.

Si se ingresa por error algún código que no existe la aplicación envía un mensaje de error informando que no existe el número. (Véase Figura 49).

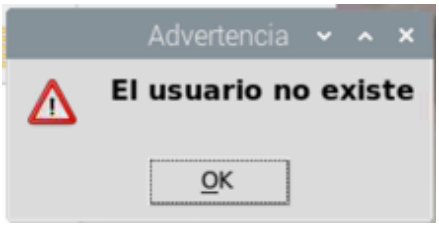


Figura 49. Advertencia de código errado.

- ***El sistema no obtiene la posición inicial o final del usuario***: puede suceder que el sistema no inicie correctamente o la unidad GPS no se encuentra en línea y transmitiendo, como consecuencia, la aplicación no

puede obtener la ubicación del usuario y no se puede realizar el cálculo de la tarifa. Para dar solución a este problema, se implementó una función que actualiza las ubicaciones faltantes cuando sea necesario, con el fin de siempre contar con los datos para realizar el cálculo de la tarifa.

9. PRUEBAS

En este capítulo se describe el proceso de pruebas que se realizó durante el desarrollo del proyecto. Se describe el plan de pruebas, se especifican los casos de pruebas y se reportan los resultados.

Desde las primeras fases de este proyecto, su enfoque fue el sector del transporte público intermunicipal de pasajeros. Pero, a raíz de la situación que actualmente sucede en el mundo por causa de la pandemia de covid-19, no se pudo realizar las pruebas de campo como estaban previstas en el cronograma de trabajo. Desde el inicio de aislamiento preventivo obligatorio. El Gobierno Nacional en conjunto con el Ministerio de Salud tomaron medidas restrictivas con el fin de retardar el contagio del virus en nuestro país. En este orden de ideas, el transporte de pasajeros se suspendió en todas sus formas, siendo el transporte local e intermunicipal el más golpeado por ser un potencial foco de propagación.

Según el artículo 4 del decreto 482 del 26 de marzo de 2020, se indica las condiciones para prestar el servicio de transporte intermunicipal.

Articulo 4. *Transporte de Pasajeros por Carretera • Intermunicipal. Durante el estado emergencia económica, social y ecológica y el aislamiento preventivo obligatorio, se permite operar el servicio público de transporte automotor en la modalidad de pasajeros por carretera - intermunicipal, con fines acceso o de prestación de servicios salud; y a personas que requieran movilizarse y sean autorizadas en los términos del Decreto 457 de 22 de marzo de 2020*[51][52].

Cabe recordar que los decretos [51][52], se expidieron desde el inicio del aislamiento preventivo obligatorio y que posteriormente, el Gobierno nacional inició con la reactivación de algunos sectores de la economía, entre ellos el transporte público local o masivo con restricciones, estas según el decreto 575 del 15 de abril de 2020 [53] que ordena: *que posterior a la medida de aislamiento obligatorio*

general, y mientras se mantenga la emergencia sanitaria, se tendrá un período en el cual se requiere mantener el confinamiento para algunos sectores de nuestra sociedad, así como también, la población de adultos mayores y estudiantes deberán acatar las medidas actuales para contener, mitigar y prevenir la expansión del coronavirus COVID-19. Se restringe la oferta de servicios en un nivel de ocupación no superior al 35% de cada vehículo de servicio de transporte masivo, lo que implica a los sistemas incurrir en costos por oferta de servicios muy superiores a los ingresos por venta de pasajes. Esta restricción también se extendió al transporte intermunicipal y hasta el momento, el transporte intermunicipal se encuentra suspendido a nivel nacional y solo se presta el servicio salvo a ciertas excepciones y a una distancia no mayor a 40km, es decir, abarca municipios cercanos del área metropolitana de ciudades capitales.

Aunque el servicio se presta a media marcha, algunas empresas de transporte público intermunicipal no permiten el uso de sus vehículos por el riesgo de contagio, tanto de los usuarios, como de quien desea realizar las pruebas, sumado a esto, la grave crisis que afecta al sector hace prácticamente imposible obtener algún permiso para utilizar los vehículos de transporte intermunicipal de pasajeros. Adicionalmente, algunas personas no permiten realizar imágenes de su rostro, otras solicitan soporte con documentación donde se especifique el uso que se dará a sus imágenes y hasta sienten temor de contagio, a raíz de la situación por la que atravesamos actualmente.

Por todo lo anterior se realiza un plan de pruebas que permitió realizar una serie de simulaciones donde se probó todos los módulos del producto de software y con los resultados de estas fue necesario realizar algunos ajustes finales. En la siguiente sección se describe el plan de pruebas aplicado:

9.1. Información del proyecto

Proyecto	Transass
Fecha de preparación	2020-07-06
Cliente	Jair Vidal
Patrocinador principal	Universidad javeriana
Gerente / Líder de proyecto	Jair Vidal
Gerente / Líder de pruebas de software	Jair Vidal

9.2. Resumen

En esta sección, se genera el plan de pruebas para los componentes del software, que permitirá encontrar sus defectos, corregirlos y realizar pruebas de regresión para su posterior puesta en producción. Las pruebas que se realizarán, se aplicarán solo a los siguientes componentes: identificación, cálculo y notificación. De igual manera, el plan de pruebas contará con las siguientes restricciones:

- **Tiempo**: Se espera que este proceso dure alrededor de 15 días (2 semanas) durante este periodo, se aplicarán las pruebas durante jornadas de aproximadamente 5 horas diarias.
- **Costo**: El plan de pruebas hasta el momento no se ha considerado un costo adicional, pues se realizará con personal propio.
- **Alcance**: Con base a las restricciones que se presenten en este documento.

9.3. Alcance de las pruebas:

El presente plan de pruebas solo se realizará para los componentes que se han desarrollado, los cuales se enunciaran a continuación:

- Módulo de identificación de usuario: permite capturar, detectar e identificar al usuario por medio de video streaming o fotos capturadas previamente.

- Módulo de cálculo: permite realizar el cálculo de las tarifas.
- Módulo de notificación: permite generar mensajes informativos para el usuario.

9.4. Funcionalidades aprobar:

9.4.1.1. Módulo de identificación de usuario:

- Captura de imagen
- Detección de rostros
- Reconocimiento

9.4.1.2. Módulo de tarifa

- Cálculo de tarifa

9.4.1.3. Módulo notificación

- Mensajes informativos
- Eliminación de datos de usuario

9.5. Pruebas de regresión:

Después de realizar las pruebas funcionales de cada uno de los componentes antes descritos, se informará al desarrollador para que realicen los ajustes del caso y, después de esto, se procederá a aplicar nuevamente las pruebas.

9.6. Criterios de salida:

9.6.1. Entregables:

Después de ejecutar este plan de pruebas se adicionará a este documento:

- Casos de pruebas aplicados con sus resultados.

- Reporte de hallazgos de las pruebas.

9.7. Recursos:

En esta sección se enuncian, los recursos de hardware y software que se utilizaran en las pruebas.

9.7.1. Hardware

Para realizar las pruebas se requiere un Raspberry pc con las siguientes características:

marca	Raspberry	
Modelo	3B+	4
Procesador:	ARM	ARM octacore
Memoria	1GB	4 GB
Disco	Micro SD 32 GB	Micro SD de 32 GB
Red	Wifi o cableada	

9.7.2. Software

La aplicación que se requiere probar, exige el siguiente software:

Sistema operativo	Raspbian basado en distribución Linux
Lenguaje de programación	Python 3.7

9.8. Estrategia de pruebas

Se requiere verificar el comportamiento del software en un ambiente muy cercano a la realidad de una operación normal de transporte. De esta manera se plantea, que las pruebas se aplicarán a los componentes antes mencionados y con los siguientes tipos de pruebas:

Nivel de prueba	Tipo de prueba
Pruebas de integración	Pruebas funcionales
Pruebas de sistema	Pruebas funcionales

9.8.1. Planificación

9.8.1.1. Procedimiento de las pruebas

El proceso de pruebas se realizará sobre la plataforma de hardware descrita en la sección de recursos y se creará una base de datos o conjuntos de archivos de imagen en formatos *.jpg y *.png que se obtuvieron de los sitios: www.gettyimages.es, www.pinterest.com y otras de redes sociales.
Se seleccionarán imágenes de rostros en múltiples posturas, con diferentes cambios de iluminación y variados colores de fondo.

Se elabora un resumen con los casos de pruebas a aplicar donde se describe el módulo, la funcionalidad y el tiempo de aplicación.

Se adaptará el sistema creando una interfaz para cargar los archivos que se utilizarán en las diferentes pruebas, se visualizará el proceso de carga y resultado de cada ejecución.

Cada caso de prueba se especificará en un cuadro donde se describe el propósito de los resultados esperados y reales

Finalizando el documento, se reportan los hallazgos y se presentarán tablas comparativas con los resultados de las pruebas.

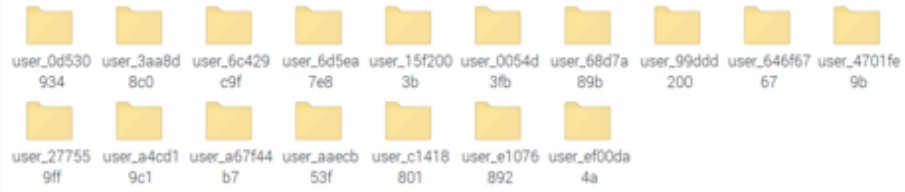
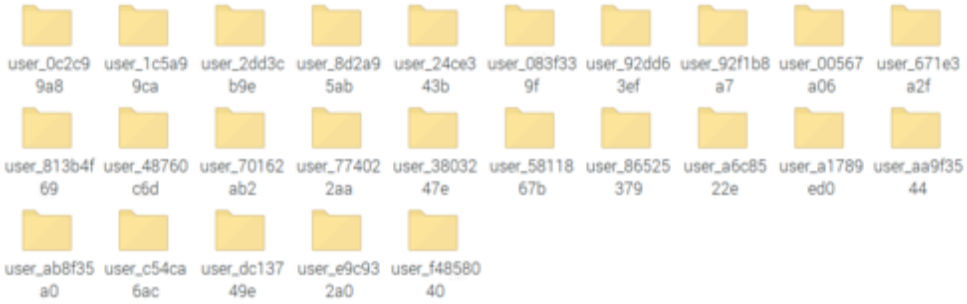
9.8.1.2. **Matriz de responsabilidades**

Tabla 21. Matriz de responsabilidades

Código	Prueba	Módulo	Funcionalidad	Nivel	Tipo	Tiempo	Recursos	Responsable
CP-001	Conjuntos de usuarios rangos de 20,30 y 50 ingresando al vehículo simulando el inicio del viaje para un automotor de esa capacidad.	Detección	Captura y Detección	Sistema	Funcional	1 hora	Base de datos de imágenes libres.	Jair Vidal
CP-002	Descenso de 20 usuarios del vehículo, distribuido en: un conjunto de 10 que simula el fin de un viaje con distancias completas *diferentes* notificando la tarifa que deben pagar y un conjunto de 10 que simula el fin de viaje con distancias completas *iguales* notificando la tarifa que deben pagar.	Identificación	Captura, detección y reconocimiento					
CP-003	Descenso de 30 usuarios del vehículo simulando el fin de viaje con distancias completas iguales.							
CP-004	Descenso de 50 de usuarios distribuido en dos conjuntos: un conjunto que simula el fin de viaje con distancias incompletas y otro que simula el fin de viaje con distancias completas.							
CP-005	Ingreso y descenso de hasta 10 usuarios sin captura inicial de imagen.							
CP-006	Ingreso de hasta 20 usuarios, distribuidos en dos conjuntos: uno simulando el ingreso sin la captura de imagen inicial y otro simulando que ingresan con la captura de imagen inicial.							
CP-007	Descenso de hasta 30 usuarios, distribuidos en: un conjunto de 15 de estos que descienden del vehículo sin captura de imagen final y un conjunto de 15 de estos, que descienden con su captura de imagen final.							
CP-008	Conjunto de 10 usuarios que inician el viaje en un mismo punto y finaliza en distancias completas diferentes, calculando la distancia y notificando el valor de la tarifa.	Tarifa	Cálculo de distancia					
CP-009	Conjunto de 10 usuarios distribuidos así: 5 usuarios sin identificar se les notifique el valor de la tarifa. 5 usuarios identificados se les notifique el valor de la tarifa.		Cálculo de tarifa con y sin reconocimiento.					
CP-010	Conjunto de 10 usuarios que ingresan al vehículo con captura de imagen del rostro y huella dactilar	Identificación	Captura, detección y reconocimiento por huella dactilar.					
CP-011	Ingreso de 10 usuarios que ingresan al vehículo con captura de imagen y huella dactilar y descenso del mismo conjunto sin captura de imagen, pero con huella dactilar.							
CP-012	Conjunto de 10 usuarios que ingresan al vehículo con captura de imagen inicial y huella dactilar distinta del usuario real y descenso de los 10 usuarios sin la captura de imagen final y la huella dactilar propia del usuario.							

9.8.1.3. **Casos de prueba**

Tabla 22. Caso de prueba 001

	FORMATO CASO DE PRUEBA	**Elaborado por**: Jair Vidal	**Aprobado por**:
Identificador	CP - 001	**Versión**	1.0
Responsable:	Jair Vidal		
Nombre caso de prueba:	Ingreso de usuarios		
Módulo		**Sub - módulo**	
Identificación de usuario		No aplica	
Formulario			
No aplica			
Descripción de la prueba			
Conjuntos de usuarios con rangos de 20,30 y 50 ingresando al vehículo simulando el inicio del viaje para un automotor de esa capacidad. Se ingresa cada 1 segundos un nuevo usuario. El sistema captura una imagen de una base de datos de imágenes por cada usuario que ingrese, se asigna una ubicación aleatoria y se contará la cantidad de usuarios hasta llegar el límite permitido.			
Propósito:			
Validar el nivel de acierto del detector de rostros al momento del ingreso de los usuarios al sistema.			
Resultado esperado:			
El sistema obtiene la imagen de una base de datos de imágenes detecta el rostro en la imagen, se crea una carpeta y dentro de esta se adiciona la imagen que se ha extraído de la base de datos. En total deberá existir 20 carpetas y cada una con 1 imagen.			
Resultado real:			
El sistema permitió el ingreso de solo 17 de las 20 imágenes que se habían suministrado para la prueba. 3 imágenes no fueron tenidas en cuenta debido a la posición del rostro al momento de la captura. 		
El sistema solo construye 25 carpetas que corresponden a los usuarios que ingresan al sistema y son reconocidos por el software. En otros casos la posición del rostro no permite la detección correcta. 		
El sistema solo construye 48 carpetas que corresponden a los usuarios que ingresan al sistema y son reconocidos por el software. En otros casos la posición del rostro o tal vez la iluminación no permite la detección correcta.			

Error:

No genera ningún error

Imagen:

Grupo de 20 imágenes

Grupo de 30 imágenes

Grupo de 50 imágenes


Ingresos
INGRESAN: 48
NO INGRESAN: 2
TOTAL: 50
OK
104.jpg
105.jpg
106.jpg
107.jpg
108.jpg
109.jpg
110.jpg
111.jpg
112.jpg
114.jpg
115.jpg
116.jpg
117.jpg
118.jpg
119.jpg
120.jpg
122.jpg
123.jpg
124.jpg
125.jpg
127.jpg
128.jpg
129.jpg
130.jpg
131.jpg
132.jpg
133.jpg
134.jpg
135.jpg
136.jpg
137.jpg
138.jpg
139.jpg
140.jpg
141.jpg
142.jpg

Tabla 23. Caso de prueba 002

	FORMATO CASO DE PRUEBA	**Elaborado por**: Jair Vidal	**Aprobado por**:
Identificador	CP - 002	**Versión**	1.0
Responsable:	Jair Vidal		
Nombre caso de prueba:	Ingreso de usuarios		
Módulo		**Sub - módulo**	
Identificación de usuario		No aplica	
Formulario			
No aplica			
Descripción de la prueba			
Descenso de 20 usuarios del vehículo, distribuido en: un conjunto de 10 que simula el fin de un viaje con distancias completas *diferentes* notificando la tarifa que deben pagar y un conjunto de 10 que simula el fin de viaje con distancias completas *iguales* notificando la tarifa que deben pagar.			
Propósito:			
Validar el comportamiento del sistema cuando los usuarios ingresan todos en un mismo punto y descienden en puntos diferentes, se requiere verificar el desempeño de las siguientes funcionalidades: captura, detección, reconocimiento facial, cálculo de distancia y notificación de la tarifa.			
Resultado esperado:			
Descenso de 10 usuarios con distancias completas iguales notificando la tarifa que debe pagar Descenso de 10 usuarios con distancias completas diferentes notificando la tarifa que debe pagar.			
Resultado real:			
a) Descenso de 10 usuarios con distancias completas *iguales* notificando la tarifa que debe pagar: en este caso solo se identifican 7 y los 3 restantes se notifican como desconocidos. En cuanto al cálculo de la tarifa se notifican todos los 10. Para efectos de esta simulación se realiza un viaje desde el punto de inicio “Cali” donde todos los usuarios ingresan y finalizan en el punto final “Palmira” donde todos descienden. b) Descenso de 10 usuarios con distancias completas *diferentes* notificando la tarifa que deben pagar: en este caso se identifican 10. En cuanto al cálculo de la tarifa se notifica todos los 10. Para efectos de esta simulación se realiza un viaje desde el punto de inicio “Cali” donde todos los usuarios ingresan y finaliza en los puntos finales “Palmira”, “Buga”, “Tuluá”, “Bugalagrande”, “La Paila” donde cada uno desciende.			
Error:			
No se reporta			
Imagen:			

Figura 50. Conjunto(a) de datos CP-002

Figura 51. Conjunto(b) de datos CP-002.

Tabla 24. Caso de prueba 003

	FORMATO CASO DE PRUEBA	**Elaborado por**: Jair Vidal	**Aprobado por**:
Identificador	CP - 003	**Versión**	1.0
Responsable:	Jair Vidal		
Nombre caso de prueba:	Ingreso de usuarios		
Módulo		**Sub - módulo**	
Identificación de usuario		No aplica	
Formulario			
No aplica			
Descripción de la prueba			
Descenso de 30 usuarios del vehículo simulando el fin de viaje con distancias completas iguales.			
Resultado esperado:			
Descenso de 30 usuarios del vehículo correctamente.			
Propósito:			
Validar el comportamiento del sistema cuando los usuarios ingresan todos en un mismo punto y descienden todos en un mismo punto de destino. Se requiere verificar el desempeño de las siguientes funcionalidades: captura, detección, reconocimiento facial, cálculo de la distancia y notificación de la tarifa.			
Resultado real:			
En total 20 usuarios fueron notificados con la tarifa, mientras que 10 de estos no se pudieron notificar puesto que algunos se encontraban en posiciones que no dejaban ver su rostro (véase Figura 52). En cuanto a la tarifa, se notifica a los que fueron identificados. Para efectos de esta simulación se realiza un viaje desde el punto de inicio “Cali” donde todos los usuarios ingresan y finaliza en el punto final “Palmira” donde todos descienden.			
Error:			
No se reporta			
Imagen:			

Figura 52. Conjunto de datos CP-003.

Tabla 25. Caso de prueba 004

	FORMATO CASO DE PRUEBA	**Elaborado por**: Jair Vidal	**Aprobado por**:
Identificador	CP - 004	**Versión**	1.0
Responsable:	Jair Vidal		
Nombre caso de prueba:	Ingreso de usuarios		
Módulo		**Sub – módulo**	
Identificación de usuario		No aplica	
Formulario			
No aplica			
Descripción de la prueba			
Descenso de 50 de usuarios distribuido en dos conjuntos: un conjunto que simula el fin de viaje con distancias incompletas y otro que simula el fin de viaje con distancias completas.			
Propósito:			
Validar el comportamiento del sistema cuando algunos usuarios ingresan en un mismo punto, pero el sistema no captura la posición actual y descienden en un mismo punto, se requiere verificar el desempeño de las siguientes funcionalidades: captura, detección, reconocimiento facial, cálculo de la distancia y notificación de la tarifa.			
Resultado esperado:			
Descenso de los 50 pasajeros notificando la tarifa			
Resultado real:			
En un conjunto de 25 usuario con distancias completas y capturas en su gran mayoría en buenas condiciones, descienden 22, de los cuales 3 no son identificados por el clasificador. En el conjunto de 25 usuarios con distancias incompletas y capturas en su gran mayoría en buenas condiciones, descienden 23, de los cuales 2 no son identificados por el clasificador, pero se notifica la tarifa.			
Error:			
No se reporta			
Imagen:			

Figura 53. Conjunto de datos CP-004.

Tabla 26. Caso de prueba 005

	FORMATO CASO DE PRUEBA	**Elaborado por**: Jair Vidal	**Aprobado por**:
Identificador	CP - 005	**Versión**	1.0
Responsable:	Jair Vidal		
Nombre caso de prueba:	Ingreso de usuarios		
Módulo		**Sub - módulo**	
Identificación de usuario		No aplica	
Formulario			
No aplica			
Descripción de la prueba			
Ingreso y descenso de hasta 10 usuarios sin captura inicial de imagen.			
Propósito:			
Validar el comportamiento del sistema cuando la captura de imagen inicial no existe. Se requiere verificar el desempeño de las siguientes funcionalidades: reconocimiento facial, cálculo de la distancia y notificación de la tarifa.			
Resultado esperado:			
Ingreso y descenso de todos los usuarios			
Resultado real:			
Descienden todos los usuarios y se notifica la tarifa			
Error:			
Se reporta inicialmente un fallo al detectar el rostro, puesto que no encuentra la imagen inicial. Se reporta fallo al realizar el reconocimiento, pues no encuentra la imagen inicial.			
Imagen:			
Solo se adiciona con un conjunto imágenes de salida, puesto que, al no tener captura inicial de imagen, no se requiere el conjunto inicial.			

Figura 54. Conjunto de datos CP-005.

Tabla 27. Caso de prueba 006

	FORMATO CASO DE PRUEBA	**Elaborado por**: Jair Vidal	**Aprobado por**:
Identificador	CP – 006	**Versión**	1.0
Responsable:	Jair Vidal		
Nombre caso de prueba:	Ingreso de usuarios		
Módulo		**Sub - módulo**	
Identificación de usuario		No aplica	
Formulario			
No aplica			
Descripción de la prueba			
Ingreso de hasta 20 usuarios, distribuidos en dos conjuntos: uno simulando el ingreso sin la captura de imagen inicial y otro simulando que ingresan con la captura de imagen inicial.			
Propósito:			
Validar el comportamiento del sistema cuando la captura de imagen inicial no existe parcialmente. Se requiere verificar el desempeño de las siguientes funcionalidades: reconocimiento facial, cálculo de la distancia y notificación de la tarifa.			
Resultado esperado:			
Descenso completo de todos los usuarios			
Resultado real:			
En un conjunto de 10 usuarios sin captura inicial, descienden 10 y se notifica la tarifa. En un conjunto de 10 usuarios con captura inicial, descienden 8 y se notifica la tarifa, los 2 restantes no son identificados, pero se notifica la tarifa.			
Error:			
Se reporta inicialmente un fallo al detectar el rostro, puesto que no encuentra la imagen inicial. Se reporta fallo al realizar el reconocimiento pues no encuentra la imagen inicial.			
Imagen:			
En la primera parte de la prueba solo se utiliza el siguiente conjunto para identificar a los usuarios. En la segunda parte de la prueba se usa el mismo conjunto para ingresar e identificar. 			

Figura 55. Conjunto de datos CP-006.

Tabla 28. Caso de prueba 007

	FORMATO CASO DE PRUEBA	**Elaborado por**: Jair Vidal	**Aprobado por**:
Identificador	CP – 007	**Versión**	1.0
Responsable:	Jair Vidal		
Nombre caso de prueba:	Ingreso de usuarios		
Módulo		**Sub - módulo**	
Identificación de usuario		No aplica	
Formulario			
No aplica			
Descripción de la prueba			
Descenso de hasta 30 usuarios, distribuidos en: un conjunto de 15 usuarios que descienden del vehículo sin captura de imagen final y un conjunto de 15 usuarios, que descienden con su captura de imagen final.			
Propósito:			
Validar el comportamiento del sistema cuando la captura de imagen final no existe. Se requiere verificar el desempeño de las siguientes funcionalidades: reconocimiento facial, cálculo de la distancia y notificación de la tarifa.			
Resultado esperado:			
Descenso de todos los usuarios			
Resultado real:			
En un conjunto de 15 usuarios con captura de imagen final, descienden 8 y se notifica la tarifa y los restantes 7 no son identificados, pero se notifica la tarifa. En un conjunto de 15 usuarios sin captura de imagen final, descienden 12 que fueron identificados en forma manual y se notifica la tarifa, los 3 restantes no son identificados, pero se notifica la tarifa.			
Error:			
Se reporta fallo al realizar el cálculo de la tarifa, pues no reporta si el usuario fue o no identificado.			
Imagen:			

Figura 56. Conjunto de datos CP-007

Tabla 29. Caso de prueba 008

	FORMATO CASO DE PRUEBA	**Elaborado por**: Jair Vidal	**Aprobado por**:
Identificador	CP – 008	**Versión**	1.0
Responsable:	Jair Vidal		
Nombre caso de prueba:	Ingreso de usuarios		
Módulo		**Sub - módulo**	
Identificación de usuario		No aplica	
Formulario			
No aplica			
Descripción de la prueba			
Conjunto de 10 usuarios que inician el viaje en un mismo punto y finaliza en distancias completas diferentes, calculando la distancia y notificando el valor de la tarifa.			
Propósito:			
Validar el comportamiento del sistema cuando los puntos de descenso de los usuarios son distintos. Se requiere verificar el desempeño de las siguientes funcionalidades: cálculo de la distancia y notificación de la tarifa.			
Resultado esperado:			
Conjunto de 10 usuarios identificados y notificados de la tarifa.			
Resultado real:			
Del conjunto de imágenes de 10 usuarios, a todos se les notifica la tarifa, pero se identifican 7; los restantes presentaron problemas para ser identificados.			
Error:			
No se reporta			
Imagen:			

Figura 57. Conjunto de datos CP-008.

Tabla 30. Caso de prueba 009

	FORMATO CASO DE PRUEBA	**Elaborado por**: Jair Vidal	**Aprobado por**:
Identificador	CP – 009	**Versión**	1.0
Responsable:	Jair Vidal		
Nombre caso de prueba:	Ingreso de usuarios		
Módulo		**Sub - módulo**	
Identificación de usuario		No aplica	
Formulario			
No aplica			
Descripción de la prueba			
Conjunto de 10 usuarios distribuidos así: 5 usuarios sin identificar, a quienes se les notifique el valor de la tarifa. 5 usuarios identificados, a quienes se les notifique el valor de la tarifa.			
Propósito:			
Validar el comportamiento del sistema cuando los usuarios son y no son identificados. Se requiere verificar el desempeño de las siguientes funcionalidades: reconocimiento facial, cálculo de la distancia y notificación de la tarifa.			
Resultado esperado:			
Se notifica la tarifa al conjunto de 10 usuarios			
Resultado real:			
Al conjunto de 5 usuario se notifica la tarifa calculada desde el lugar donde inicio el viaje hasta donde descendió, omitiendo la identificación debido a que no se logró. El segundo conjunto de usuarios se notifica correctamente.			
Error:			
Se reporta fallo al realizar el cálculo de la tarifa, pues reporta que el usuario no fue identificado.			
Imagen:			

Figura 58. Conjunto de datos CP-009

Tabla 31. Caso de prueba 010

	FORMATO CASO DE PRUEBA	**Elaborado por**: Jair Vidal	**Aprobado por**:
Identificador	CP – 010	**Versión**	1.0
Responsable:	Jair Vidal		
Nombre caso de prueba:	Ingreso de usuarios combinando foto y huella dactilar		
Módulo		**Sub - módulo**	
Identificación de usuario		No aplica	
Formulario			
No aplica			
Descripción de la prueba			
Conjunto de 10 usuarios que ingresan al vehículo con captura de imagen del rostro y huella dactilar			
Propósito:			
Validar el comportamiento del sistema cuando éste tiene conectado el escáner de huella dactilar. Se requiere verificar el desempeño de las siguientes funcionalidades: escáner de huella, captura de imagen, reconocimiento facial, cálculo de la distancia y notificación de la tarifa.			
Resultado esperado:			
El ingreso y descenso de todo el grupo de usuarios sin novedades			
Resultado real:			
Del conjunto de 10 usuarios ingresan todos con su captura de imagen y huella dactilar, pero al momento de realizar el reconocimiento facial, solo se pueden identificar 8, los restantes presentan problemas con la captura de imagen final, los 2 usuarios que no se identifican se les notifica un valor de tarifa errado.			
Error:			
Se reporta un fallo cuando no se encuentra conectado el escáner de huella. Se reporta un fallo al calcular la distancia debido a la falta de uno de los valores de la posición del usuario.			
Imagen:			

Figura 59. Conjunto de datos CP-010

Tabla 32. Caso de prueba 011

	FORMATO CASO DE PRUEBA	**Elaborado por**: Jair Vidal	**Aprobado por**:
Identificador	CP – 011	**Versión**	1.0
Responsable:	Jair Vidal		
Nombre caso de prueba:	Ingreso de usuarios combinando foto con huella dactilar		
Módulo		**Sub - módulo**	
Identificación de usuario		No aplica	
Formulario			
No aplica			
Descripción de la prueba			
Ingreso de 10 usuarios que ingresan al vehículo con captura de imagen y huella dactilar y descenso del mismo conjunto sin captura de imagen, pero con huella dactilar.			
Propósito:			
Validar el comportamiento del sistema cuando el escáner se encuentra conectado y no se logra la captura de la imagen final. Se requiere verificar el desempeño de las siguientes funcionalidades: escáner de huella dactilar, reconocimiento facial, cálculo de la distancia y notificación de la tarifa.			
Resultado esperado:			
Identificación correcta de todos los usuarios combinando los dos métodos de identificación.			
Resultado real:			
Del conjunto de 10, el sistema identifica 6 con foto y huella, 2 con foto solamente y los restantes no son identificados y no reconoce las huellas. El cálculo de la tarifa para 2 usuarios presentó problemas dando resultados errados.			
Error:			
Se reporta un fallo cuando no se encuentra la imagen final. Se reporta un fallo cuando no se encuentra conectado el escáner de huella.			
Imagen:			

Figura 60. Conjunto de datos CP-011.

Tabla 33. Caso de prueba 012

	FORMATO CASO DE PRUEBA	**Elaborado por**: Jair Vidal	**Aprobado por**:
Identificador	CP – 012	**Versión**	1.0
Responsable:	Jair Vidal		
Nombre caso de prueba:	Ingreso de usuarios combinando foto y huella dactilar		
Módulo		**Sub - módulo**	
Identificación de usuario		No aplica	
Formulario			
No aplica			
Descripción de la prueba			
Conjunto de 10 usuarios que ingresan al vehículo con captura de imagen inicial y huella dactilar distinta del usuario real y descenso de los 10 usuarios sin la captura de imagen final y la huella dactilar propia del usuario.			
Propósito:			
Validar el comportamiento del sistema cuando se captura la imagen inicial del usuario y el escáner se encuentra conectado y se captura la huella de otro usuario. Se requiere verificar el desempeño de las siguientes funcionalidades: escáner de huella dactilar, reconocimiento facial, cálculo de la distancia y notificación de la tarifa.			
Resultado esperado:			
Identificación correcta de todos los usuarios combinando los dos métodos de identificación.			
Resultado real:			
Ningún usuario fue identificado combinando los dos métodos de identificación, dado que la huella no corresponde al usuario, se debe ajustar el software para superar esta prueba.			
Error:			
Se reporta un fallo cuando no se encuentra la imagen final. Se reporta un fallo cuando no se encuentra conectado el escáner de huella.			
Imagen:			

Figura 61. Conjunto de datos CP-012.

Tabla 34. Reporte de hallazgo iteración 1

# CP	Fecha	Hallazgos	Resultados	Recomendaciones
CP - 001	21/07/2020	• El sistema no permite detectar al usuario al inicio	No exitoso	• Ajustar el componente de detección.
CP – 002	24/07/2020	• El sistema no permite que los usuarios que no puedan ser identificados puedan ser notificados de la tarifa.	No exitoso	• Ajustar el componente de identificación que permita detectar usuarios en la gran mayoría de posturas y cambios de luminosidad.
CP – 003	25/07/2020	• Algunos usuarios presentan problemas para realizar el reconocimiento facial. • El sistema No puede calcular una tarifa dentro del perímetro urbano.	No exitoso	• Establecer otros métodos para la identificación del usuario. • Ajustar el componente de tarifa que permite calcular un cobro local.
CP – 004	26/07/2020	• El sistema no puede calcular la tarifa por que hace falta uno de los dos puntos necesarios para el cálculo.	No exitoso	• Integrar componentes de hardware o software que permita obtener datos más precisos de geolocalización. • Implementar solución de copias de seguridad con el fin de mantener la información a salvo. • obtener la posición del usuario aun cuando el sistema se apague o presente fallos.
CP – 005	27/07/2020	• El sistema No permite el ingreso de usuarios si estos no poseen captura inicial de imagen.	No exitoso	• Establecer otros métodos para la identificación del usuario.
CP – 006	28/07/2020	•	No exitoso	
CP – 007	29/07/2020	• El sistema No puede calcular la tarifa pues no posee la captura final de imagen.	No exitoso	• Establecer otros métodos para la identificación del usuario. • Calcular la tarifa en casos excepcionales como falta de una de las dos capturas, usuario no identificado, huellas dactilares defectuosas o huellas dactilares que no concuerdan.
CP – 008	30/07/2020		Exitoso	
CP – 009	01/08/2020	• El sistema no puedo calcular la tarifa pues el usuario no es identificado.	No exitoso	• Establecer otros métodos para la identificación del usuario.
CP – 010	02/08/2020		Exitoso	
CP – 011	03/08/2020	• El sistema no puede identificar al usuario si la captura de imagen no existe, no es posible utilizar la huella dactilar para identificarlo.	No exitoso	• Establecer otros métodos para la identificación del usuario. • Crear un mecanismo que solicite información al usuario con el fin de identificar al usuario de forma manual.
CP – 012	04/08/2020	• El sistema no puede identificar al usuario si la huella no	No exitoso	• Crear un mecanismo que solicite información al

		corresponde a la misma la persona de la foto.		usuario con el fin de identificar al usuario de forma manual.
			CP exitoso	3
			CP No Exitoso	9
			TOTAL	**12**

Tabla 35. Reporte de hallazgos iteración 2

<table>
<tr><th># Caso</th><th>Fecha</th><th>Hallazgos</th><th>Resultado</th></tr>
<tr><td>CP - 001</td><td>21/07/2020</td><td>• Algunos usuarios presentan dificultad para el ingreso, debido a posiciones del rostro o cambios de luminosidad.
• De un conjunto de 20 usuarios con imágenes de mejor calidad se obtiene un ingreso de 85% (Véase tabla adjunta.)
• De un conjunto de 30 usuarios con imágenes de mejor calidad se obtiene un ingreso de 83% (Véase tabla adjunta.)
• De un conjunto de 50 usuarios con imágenes de mejor calidad se obtiene un ingreso de 96% (Véase tabla adjunta.)
• Con mejores condiciones de ambiente el porcentaje de ingreso se incrementa.
• El sistema permite el ingreso de todos los usuarios así no se detecte el rostro al iniciar el viaje.</td><td rowspan="13">Exitoso</td></tr>
<tr><td>CP – 002</td><td>24/07/2020</td><td rowspan="6">• El sistema permite que los usuarios que no puedan ser identificados puedan ser notificados de la tarifa utilizando identificación manual.
• El sistema puede calcular una tarifa local siempre y cuando el usuario no haya salido del perímetro urbano.
• Algunos usuarios presentan problemas para realizar el reconocimiento facial.
• El sistema actualiza las distancias incompletas adicionando la última distancia valida reportada.
• El sistema utiliza como recurso para calcular la tarifa la ubicación del usuario.
• El sistema permite el ingreso sin captura de imagen inicial</td></tr>
<tr><td>CP – 003</td><td>25/07/2020</td></tr>
<tr><td>CP – 004</td><td>26/07/2020</td></tr>
<tr><td>CP – 005</td><td>27/07/2020</td></tr>
<tr><td>CP – 006</td><td>28/07/2020</td></tr>
<tr><td>CP – 007</td><td>29/07/2020</td></tr>
<tr><td>CP – 008</td><td>30/07/2020</td><td rowspan="4">• Algunos usuarios presentan problemas para capturar la huella debido a que el escáner tiene el cristal con impurezas.
• El sistema guarda una copia de la huella del usuario</td></tr>
<tr><td>CP – 009</td><td>01/08/2020</td></tr>
<tr><td>CP – 010</td><td>02/08/2020</td></tr>
<tr><td>CP – 011</td><td></td></tr>
<tr><td>CP – 012</td><td></td><td>• El sistema puede identificar al usuario si la huella no corresponde a la misma la persona de la foto.</td></tr>
</table>

Casos exitosos	12
Casos No Exitosos	0
TOTAL	12

Tabla 36. Tabla comparativa de detección de imágenes

Datos de ingresos					
# usuarios	Características del conjunto de datos	Detectados	No detectados	Porcentaje	
20	• Excelentes condiciones de luminosidad. • Toma de rostro frontal. • Sin gafas, barba u otros.	17	3	Ingreso	85%
				No ingreso	15%
	• Buenas condiciones de luminosidad, aunque algunas imágenes opacas. • Toma frontal de rostro. • Algunos con anteojos otros sin este elemento. • Algunos con barba	15	5	Ingreso	75%
				No ingreso	25%
	• Aceptables condiciones de luminosidad, aunque algunas imágenes oscuras o con poca luz en el área del rostro. • Toma frontal del rostro, aunque algunas con tomas de lado o en otras posturas • Con anteojos, barba.	9	11	Ingreso	45%
				No ingreso	55%
		2	18	Ingreso	10%
				No ingreso	90%
30	• Excelentes condiciones de luminosidad. • Toma de rostro frontal. • Sin gafas, barba u otros.	25	5	Ingreso	83,3%
				No ingreso	13,6%
	• Buenas condiciones de luminosidad, aunque algunas imágenes opacas. • Toma frontal de rostro. • Algunos con anteojos otros sin este elemento.	19	11	Ingreso	63,3%
				No ingreso	36,6%
	• Aceptables condiciones de luminosidad, aunque algunas imágenes oscuras o con poca luz en el área del rostro. • Mas de un rostro en la foto • Toma frontal del rostro, aunque algunas con tomas de lado o en otras posturas. • Con anteojos, barba	14	16	Ingreso	46,6%
				No ingreso	53,3%
50	• Excelentes condiciones de luminosidad. • Toma de rostro frontal • Sin gafas, barba u otros	48	2	Ingreso	96%
				No ingreso	4%
	• Buenas condiciones de luminosidad, aunque algunas imágenes opacas.	37	13	Ingreso	74%

	• Toma frontal de rostro • Algunos con anteojos otros si este elemento			No ingreso	26%
	• Aceptables condiciones de luminosidad, aunque algunas imágenes oscuras o con poca luz en el área del rostro. • Mas de un rostro en la foto • Toma frontal del rostro, aunque algunas con tomas de lado o en otras posturas. • Con anteojos, barba • Con gorras o capucha	21	29	Ingreso	42%
				No ingreso	58%

Tabla 37. Tabla comparativa de reconocimiento facial de imágenes

Datos de reconocimiento					
# usuarios	Características del conjunto de datos	reconoce	No reconoce	Porcentaje	
20	• Excelentes condiciones de luminosidad. • Toma de rostro frontal. • Sin gafas, barba u otros.	16	4	Reconoce	80%
				No reconoce	20%
	• Buenas condiciones de luminosidad, aunque algunas imágenes opacas. • Toma frontal de rostro. • Algunos con anteojos otros si este elemento. • Algunos con barba.	12	8	Reconoce	60%
				No reconoce	40%
	• Aceptables condiciones de luminosidad, aunque algunas imágenes oscuras o con poca luz en el área del rostro. • Toma frontal del rostro, aunque algunas con tomas de lado o en otras posturas. • Con anteojo, barba.	6	14	Reconoce	30%
				No reconoce	70%
	• Deficientes condiciones de luz, con imágenes muy oscuras. • Con imágenes diversas, casi no se presenta captura de rostros frontal. • Muy pocas con tomas frontal del rostro, imágenes con más de una persona y con cambios de fondo. • Con anteojos, barba, accesorios.	0	20	Reconoce	0%
				No reconoce	100%
30	• Excelentes condiciones de luminosidad. • Toma de rostro frontal. • Sin gafas, barba u otros.	24	6	Reconoce	80%
				No reconoce	20%
	• Buenas condiciones de luminosidad, aunque algunas imágenes opacas.	14	16	Reconoce	46,6%

	• Toma frontal de rostro. • Algunos con anteojos otros si este elemento.			No reconoce	54,6%
	• Aceptables condiciones de luminosidad, aunque algunas imágenes oscuras o con poca luz en el área del rostro. • Mas de un rostro en la foto. • Toma frontal del rostro, aunque algunas con tomas de lado o en otras posturas. • Con anteojos, barba.	10	20	Reconoce	33,3%
				No reconoce	66,6%
50	• Excelentes condiciones de luminosidad. • Toma de rostro frontal. • Sin gafas, barba u otros.	46	4	Reconoce	92%
				No reconoce	8%
	• Buenas condiciones de luminosidad, aunque algunas imágenes opacas. • Toma frontal de rostro. • Algunos con anteojos otros si este elemento.	31	19	Reconoce	62%
				No reconoce	38%
	• Aceptables condiciones de luminosidad, aunque algunas imágenes oscuras o con poca luz en el área del rostro. • Mas de un rostro en la foto. • Toma frontal del rostro, aunque algunas con tomas de lado o en otras posturas. • Con anteojos, barba. • Con gorras o capucha.	19	31	Reconoce	38%
				No reconoce	62%

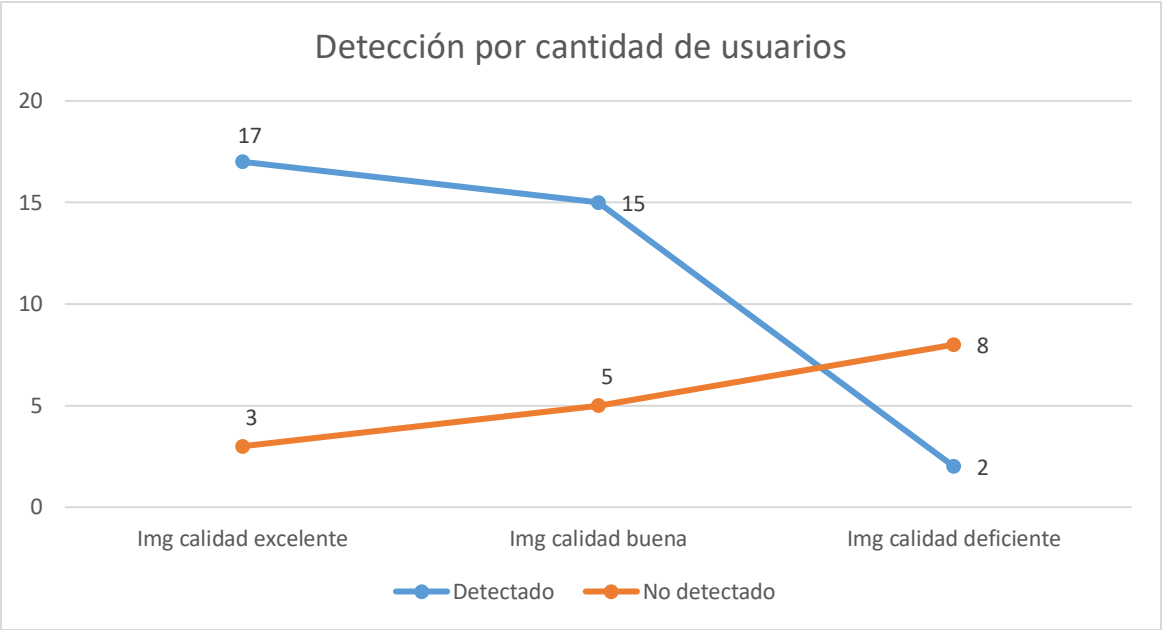


Figura 62. Gráfico 1 por cantidad de usuarios.

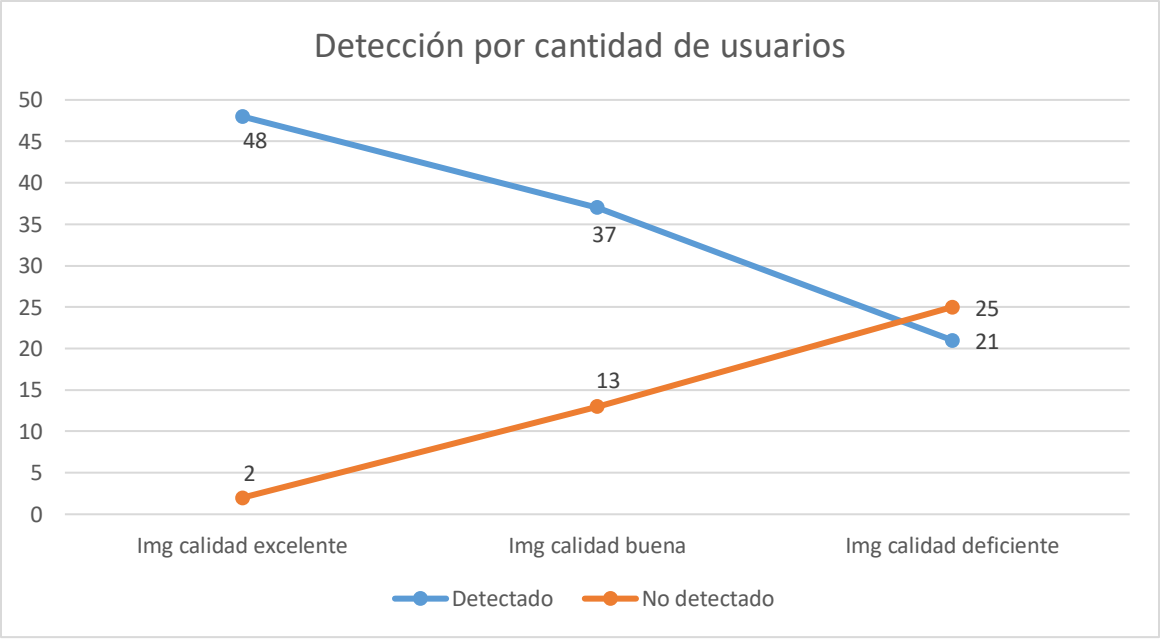


Figura 63. Gráfico 2 por cantidad de usuarios.

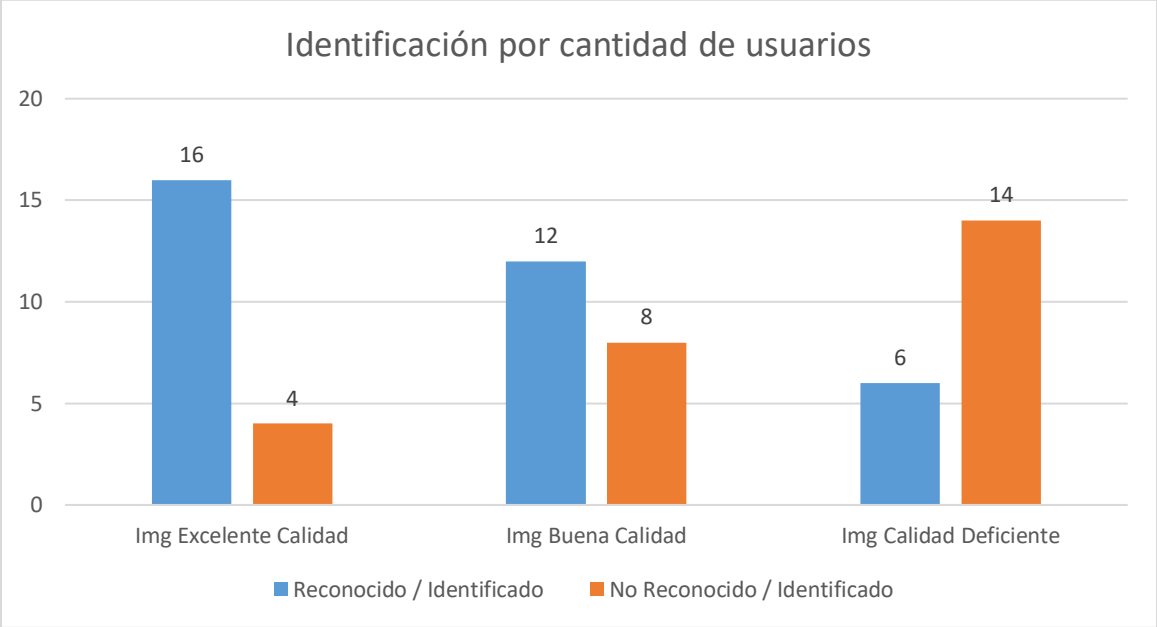


Figura 64. Gráfico 1 identificación de usuarios.

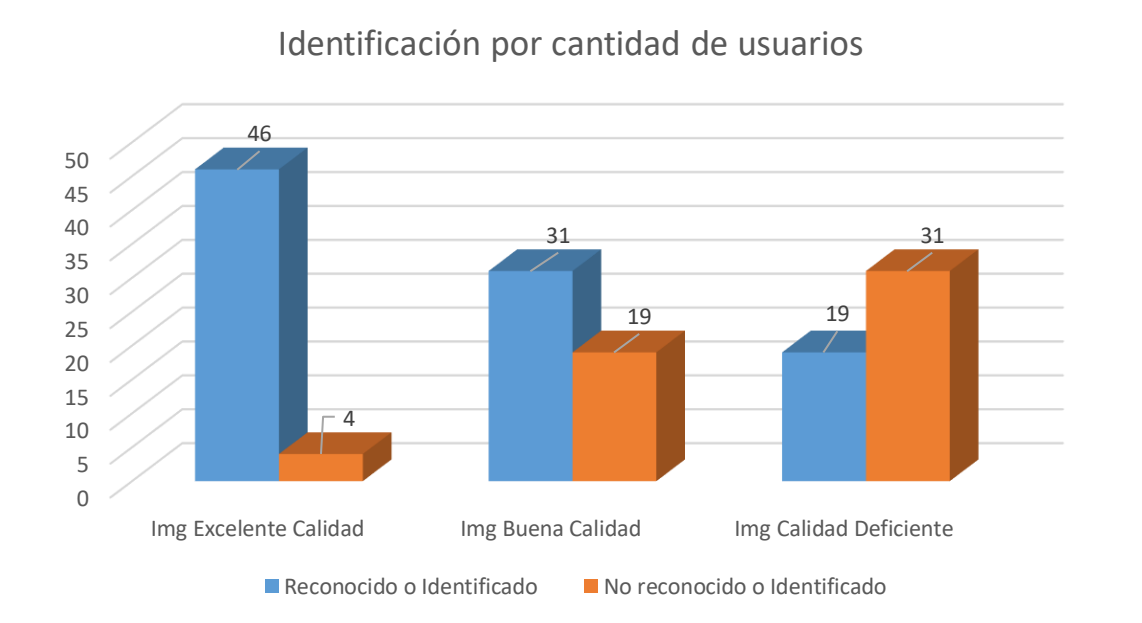


Figura 65. Gráfico 2 identificación de usuarios.

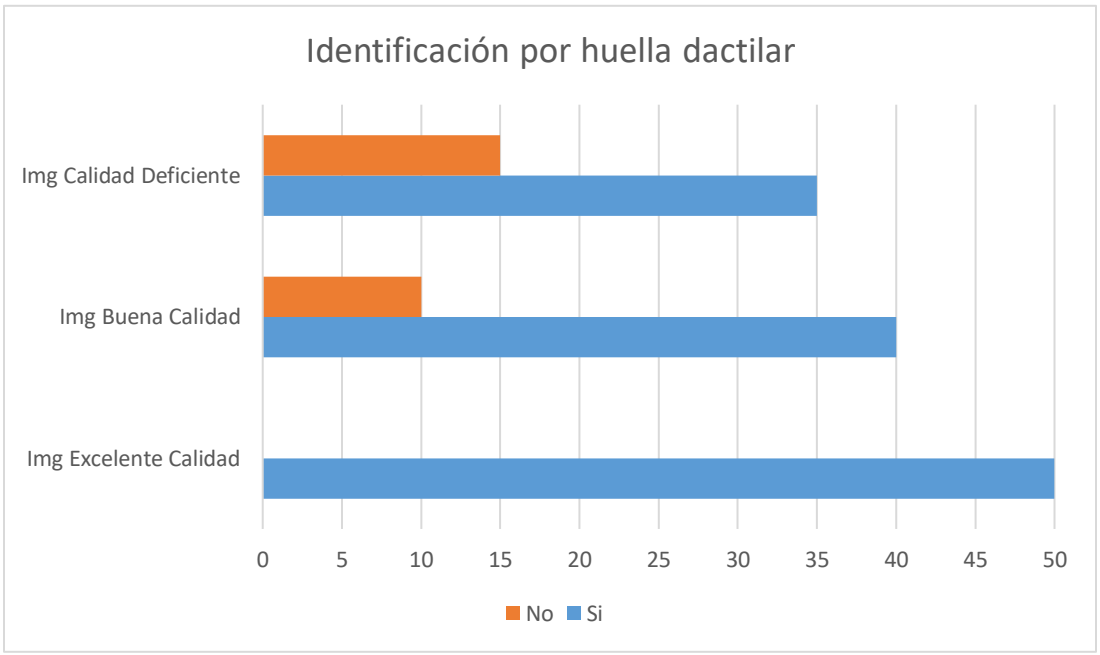


Figura 67. Gráfico identificación por huella dactilar.

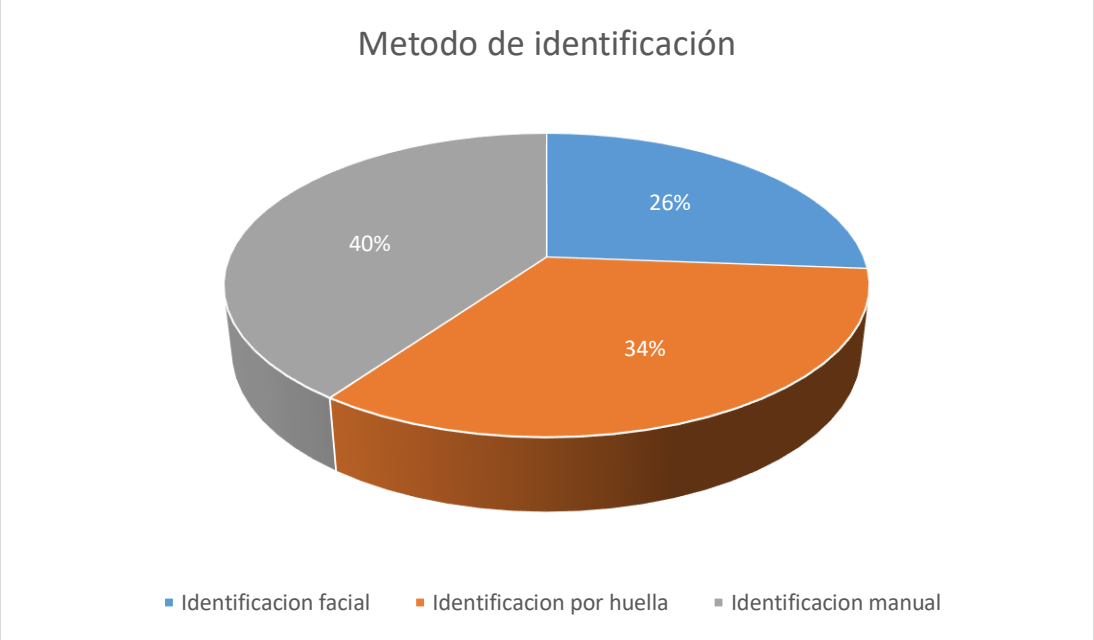


Figura 66. Gráfico Distribución por Método de identificación.

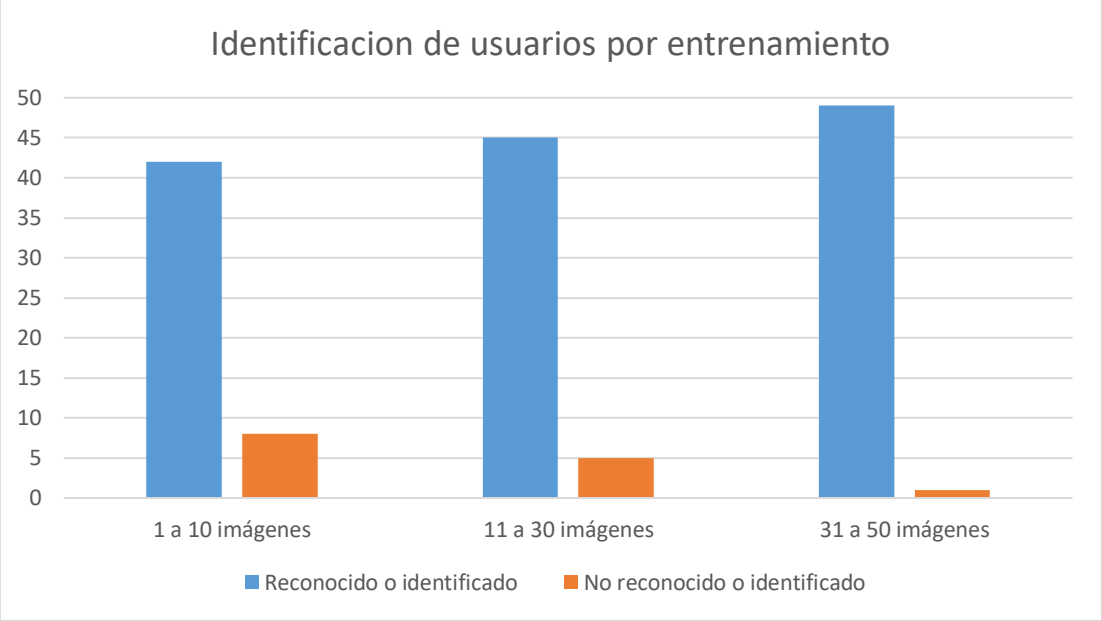


Figura 68. Gráfico por entrenamiento.

Cronograma

Tabla 38. Cronograma de actividades plan de pruebas.

Fase	Semanas														
	1								2						
	Días														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Planeación de pruebas															
Diseño de pruebas															
Elaboración de la matriz de requerimientos de prueba															
Ejecución pruebas funcionales															
Reporte de incidentes															
Pruebas de regresión															
Reporte de pruebas.															

10. CONCLUSIONES

- Este trabajo de grado se realizó dando cumplimiento a los objetivos trazados: identificar al usuario, calcular la tarifa y notificar el valor calculado, logrando construir un prototipo de software.
- El algoritmo Viola-Jones utilizado en el componente de captura de imágenes, permitió realizar una detección rápida de rostros con un porcentaje cercano al 89% de acierto, aunque la posición del rostro o los cambios iluminación afectan considerablemente el funcionamiento del algoritmo generando detecciones falsas. Mientras que el método LBPH (histograma de patrones binarios locales) permitió el reconocimiento facial con un porcentaje cercano al 80%, siendo este el más fuerte ante los cambios de iluminación. Durante el proceso de pruebas se evaluó el desempeño de los algoritmos en el ingreso y la identificación de usuarios.
- Se evidenció a través de los resultados de las pruebas que el componente *procesar imagen* encargado de identificar al usuario, presentó algunas debilidades en ambientes con condiciones de baja iluminación, posición errónea del usuario o distorsión de la imagen (simulando vibración), generando malas capturas de imagen, imágenes borrosas o imágenes vacías, aun si el nivel de confianza se disminuye, no fue posible realizar una buena identificación; por el contrario, se incrementaron las detecciones falsas o mejor llamados falsos positivos. La integración del lector de huella dactilar al sistema fortaleció el proceso de identificación e incrementó la productividad.
- En la actualidad con la crisis de pandemia que se vive, evitar el contacto con objetos es imprescindible, de ahí que tecnologías como el reconocimiento facial combinada con un módulo GPS, son herramientas que en esta coyuntura aportan información, trazabilidad y seguridad.
- Desde el inicio del proyecto hasta el final, permitió ampliar el conocimiento debido a que algunos conceptos de electrónica están fuera del alcance del

desarrollador, pero todo fue un reto que se tomó con responsabilidad y con pensamiento investigativo.

11. PRESUPUESTO

En esta sección se presentan el presupuesto del proyecto de grado.

Presupuesto General

A continuación, se presenta la tabla con el presupuesto general del proyecto.

Tabla 39. Presupuesto total del proyecto.

RUBRO	Costo (Pesos)
Personal	$ 2.000.000
Hardware	$ 575.000
Software	$ 0
Total Proyecto	$2.575.000

Presupuesto Detallado

A continuación, se presentan las tablas con los presupuestos detallados del proyecto, entre los cuales se incluyen presupuestos de personal, hardware, software, desplazamiento y papelería.

Tabla 40. Descripción detallada presupuesto Personal.

Nombre del Colaborador	Formación Académica	Función dentro del proyecto	Dedicación(horas/semana)	Costo por hora	Meses de Vinculación	Costo Total (en pesos)
Jair Hernando Vidal	Pregrado	Desarrollador	40	$5000	10	$2.000.000
Total Gastos Personal						**$2.000.000**

Tabla 41. Descripción detallada del hardware requerido.

Tipo	Descripción	Costo
Microcomputador Raspberry b+	Equipo que permitirá el desarrollo del prototipo	$ 90.000
Micro sd 30Gb	Unidad de almacenamiento	$ 30.000
Pantalla LCD compatible con rapsberry		$ 145.000
Lector de huella dactilar		$50.000
Unidad GPS		$ 80.000
batería		$ 100.000

Cámara web 720p		$ 80.000
Total Gastos Hardware		**$ 575.000**

Tabla 42. Descripción detallada del software requerido.

Tipo	Descripción	Costo
Draw.io	Herramienta que permite modelar los procesos en lenguaje UML	$ 0
BD Sqlite	Base de datos empotrada.	$ 0
Raspbian	Sistema operativo para el micro computador	$ 0
Total Gastos Software		$ 0

12. TRABAJOS FUTUROS

Se espera que este prototipo llamado Transass, transcienda más allá de la academia, esperando que se pueda mejorar o adicionar nuevas funcionalidades como:

- Añadir un módulo de transmisión de datos a la nube o a un servidor local.
- Adicionar un módulo de reportes que permita obtener información de viajes, pasajeros, etcétera.
- Aumentar un módulo estadístico que permita obtener información acerca el porcentaje de usuarios por ruta o por sector, promedio de ingresos y descensos, etcétera.
- Mejorar la interfaz de usuario

13. BIBLIOGRAFÍA

[1] M. de Transporte, “Resolución 3600 del 9 de mayo de 2001,” 2001.

[2] B. G. Han, K. T. Lim, Y. S. Chung, and S. I. Lee, “Passenger management system based on face recognition for intelligent transport vehicles,” *Int. Conf. Ubiquitous Futur. Networks, ICUFN*, no. 1, pp. 117–118, 2013, doi: 10.1109/ICUFN.2013.6614792.

[3] N. G. P. Sarathi and C. O. Otto, “Fare management system for transport corporation using face recognition based on principal component analysis,” *Proceeding IEEE Int. Conf. Green Comput. Commun. Electr. Eng. ICGCCEE 2014*, pp. 1–5, 2014, doi: 10.1109/ICGCCEE.2014.6922207.

[4] Wikipedia, “Proceso de reconocimiento facial,” *2019*. .

[5] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Dec. 2001, vol. 1, pp. I–I, doi: 10.1109/CVPR.2001.990517.

[6] Parra Barrero Eloy, “Aceleración del algoritmo de Viola-Jones mediante rejillas de procesamiento masivamente paralelo en el plano focal,” 2015, [Online]. Available: http://bibing.us.es/proyectos/abreproy/90325/fichero/TrabajoFinGrado.pdf.

[7] K. Elizabeth, R. Rodr´, and R. Baque, “Universidad T ´ Ecnica Del Norte Facultad De Ingenier´Iaingenier´Ingenier´Ia ‘Analysis and Implementation of the Viola-Jones Face Detection Algorithm.’”

[8] F. C. Crow, “Summed-Area Tables for Texture Mapping.,” *Comput. Graph.*, vol. 18, no. 3, pp. 207–212, 1984, doi: 10.1145/964965.808600.

[9] S. V. Hirvin Gonzalez, “Reconocimiento Facial utilizando Viola-Jones y Patrones Binarios,” vol. 23, p. 1, 2019, [Online]. Available: file:///C:/Users/SBryan/Downloads/126-Artículo-287-2-10-20190912.pdf.

[10] T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002, doi:

10.1109/TPAMI.2002.1017623.

[11] K. Meena and A. Suruliandi, “Local binary patterns and its variants for face recognition,” in *2011 International Conference on Recent Trends in Information Technology (ICRTIT)*, Jun. 2011, pp. 782–786, doi: 10.1109/ICRTIT.2011.5972286.

[12] J. López García, “Algoritmo para la identificación de personas basado en huellas dactilares,” pp. 1–67, 2009, [Online]. Available: http://upcommons.upc.edu/handle/2099.1/8082.

[13] D. la torre M. Ocaña Diez, “ALGORITMOS DE MATCHING ENTRE HUELLAS DACTILARES,” 2017.

[14] M. S. Sanchez, “ESTUDIO DEL RENDIMIENTO BIOMÉTRICO DE SISTEMAS DE HUELLA DACTILAR. ANÁLISIS DE DIFERENTES SENSORES Y ALGORITMOS,” *Formulacion y Eval. Proy. Agrar.*, vol. 4, pp. 1224–1234, 2015.

[15] F. Galton, “Finger Prints by Francis Galton [Facsimile Edition],” pp. 64–65, 1892, [Online]. Available: http://www.safedoc.com/BiometricBits.htm.

[16] Wikipedia, “Imagen_dedo,” 2016. .

[17] Pinterest, “imagen_huella,” 2015. .

[18] T. Electrónica, “Como funcionan los lectores de huella,” *2003*, 2003. .

[19] J. de Mendoza y Ríos, *Memoria sobre algunos métodos nuevos de calcular la longitud por las distancias lunares y explicaciones prácticas de una teoría para la solución de otros problemas de navegación*. Madrid: 1795, 1795.

[20] Wikipedia, “Ley de semirverseno,” 2017. .

[21] wikipedia, “Formula del semiverso,” 2016. .

[22] J. Martínez-Carballido, C. García-Lucero, J. M. Ramírez-Cortés, and R. Enríquez-Caldera, “FPGA based system prototype for face location and recording of people getting on a public transport bus,” *CONIELECOMP 2010 - 20th Int. Conf. Electron. Commun. Comput.*, pp. 164–168, 2010, doi: 10.1109/CONIELECOMP.2010.5440774.

[23] V.-T. Vu *et al.*, “Audio-video event recognition system for public transport security,” pp. 414–419, 2008, doi: 10.1049/ic:20060345.

[24] S. Kazi, M. Bagasrawala, F. Shaikh, and A. Sayyed, “Smart E-Ticketing System for Public Transport Bus,” *2018 Int. Conf. Smart City Emerg. Technol. ICSCET 2018*, pp. 1–7, 2018, doi: 10.1109/ICSCET.2018.8537302.

[25] G. L. Shuang Xu, Deqing Niu, Bo Tao, “Convolutional neural network based traffic sign recognition system,” no. Icsai, pp. 957–961, 2018.

[26] Optocontrol, “Contador multitarifa,” *2015*, 2019. http://www.optocontrolonline.com/html/content/esp/productos/opmtr5000.html.

[27] A. EFE, “Reconocimiento facial en china,” *2018*, 2019. https://www.efe.com/efe/america/tecnologia/china-usara-el-reconocimiento-facial-para-supervisar-a-los-conductores-de-autobus/20000036-3749174#.

[28] L. Follonier, Maximiliano; Peroni and Proyecto, “Universidad Tecnológica Nacional Proyecto Final AVSA – Asistente vehicular de seguridad,” 2018.

[29] D. Molina, “Escuela Superior Politecnica De Chimborazo,” *Infoplc.Net*, pp. 2–145, 2011, [Online]. Available: http://www.infoplc.net/files/descargas/schneider/infoplc_net_18t00436.pdf.

[30] M. F. CALLES CARRASCO, “SISTEMA INFORMÁTICO DE RECONOCIMIENTO FACIAL PARA EL REGISTRO Y CONTROL DE ASISTENCIA DE LOS SOCIOS DE LA COOPERATIVA DE TAXIS Y CAMIONETAS PUYO,” pp. 1–35, 2019, doi: .1037//0033-2909.I26.1.78.

[31] C. R. Moral, “SmartBus : Big Data & Data Science en Transporte Urbano,” *Univ. Autónoma Madrid - Tesis Master en Ing. Informática*, 2017.

[32] R. P. Foundation, “About us Foundation Raspberry,” *2000*, 2000. .

[33] R. P. Foundation, “What is Raspberry Pi,” *2002*, 2002. .

[34] R. P. Foundation, “Raspberry Pi 3 Model B,” *2012*, 2012. .

[35] R. P. Foundation, “Raspberry Pi 3 Model B+,” 2018.

[36] R. P. Foundation, “Raspberry Pi 4,” 2019. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/.

[37] Amazon, “Amazo camara RPi,” *2020*. .

[38] Logithech, “Logithech,” *2016*. .

[39] BigElectronica, “Lector de huella,” *2018*. .

[40] A.-Z. Electronics, “Modulo GPS,” *2016*. .

[41] G. P. Thompson Mike, “Raspbian,” *2019*, 2002. .

[42] Python, “What is Python?,” 2000. .

[43] Python, “What is python 3?,” 2000. https://www.pythonforbeginners.com/learn-python/what-is-python/.

[44] Pyhton, “Tkinter,” *2018*. https://docs.python.org/3/library/tk.html.

[45] B. Raschke, “PyFingerprint Library,” *2015*, 2020. https://pypi.org/project/pyfingerprint/.

[46] Opencv, “OpenCv,” 2020. https://www.filehorse.com/es/descargar-opencv/.

[47] Consortium Sqlite, “Sqlite database,” *2014*, 2020. .

[48] Microsoft, “Visual Studio Code,” *2020*, 2020. .

[49] M. Gonzalo, M. Torres, F. Ignacio, and V. Toro, “IDENTIFICACIÓN POR RADIOFRECUENCIA Pontificia Universidad Católica de Valparaíso – Chile Facultad de Ingeniería Escuela de Ingeniería Informática,” 2012.

[50] G. Solano, “Reconocimiento facial con opencv y python,” *26 de mayo de 2020*, 2020. .

[51] M. del Interior, “Decreto 00482 - 26 De Marzo De 2020,” *Pres. la República*, vol. 1, no. 1, p. 10, 2020, [Online]. Available: https://dapre.presidencia.gov.co/normativa/normativa/DECRETO 482 DEL 26 DE MARZO DE 2020.pdf.

[52] M. del Interior, “Decreto 00457 - 22 de marzo de 2020,” *Pres. la República*, vol. 1, no. 1, pp. 110–122, 2020, doi: 10.1055/s-2008-1040325.

[53] M. de Transporte, “Decreto 00575 - 15 de abril de 2020.,” *Pres. la República*, pp. 1–17, 2020, [Online]. Available: https://dapre.presidencia.gov.co/normativa/normativa/DECRETO 575 DEL 15 DE ABRIL DE 2020.pdf.