PROTOTIPO DE APLICACIÓN MÓVIL DE IDENTIFICACIÓN Y CONTEO DE PERSONAS POR MEDIO DE RECONOCIMIENTO DE IMÁGENES

JUAN SEBASTIÁN VELÁSQUEZ MANZANO

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA

FACULTAD INGENIERÍA DE SISTEMAS

TECNOLOGÍA Y SOCIEDAD

BUCARAMANGA

2021

PROTOTIPO DE APLICACIÓN MÓVIL DE IDENTIFICACIÓN Y CONTEO DE PERSONAS POR MEDIO DE RECONOCIMIENTO DE IMÁGENES

JUAN SEBASTIÁN VELÁSQUEZ MANZANO

PROYECTO DE GRADO PRESENTADO PARA OPTAR AL TÍTULO DE

INGENIERO DE SISTEMAS

DIRECTOR

JUAN SEBASTIAN CARDENAS ARENAS

INGENIERO DE SISTEMAS

CODIRECTOR

LEONARDO HERNÁN TALERO SARMIENTO

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA

FACULTAD INGENIERÍA DE SISTEMAS

GRUPO DE INVESTIGACIÓN PRISMA

TECNOLOGÍA Y SOCIEDAD

BUCARAMANGA

2021

AGRADECIMIENTOS

A los profesores y tutores que hicieron parte de todo mi proceso de formación, por sus valiosos consejos y aportes profesionales.

A mis padres que son mi ayuda incondicional para cumplir mis sueños y metas.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

La situación actual generada por la expansión del covid-19 requiere que los establecimientos como: bancos, restaurantes, tiendas, hoteles, etc, tomen soluciones que garanticen el control de aforo en tiempo real y la seguridad de los usuarios. Controlar el aforo se convierte así en una necesidad primordial para los establecimientos que puedan congregar un gran número de personas, ya que según las normativas del gobierno nacional deben disminuir su capacidad total en un 40%. Por esta razón el desarrollo de una aplicación para dispositivos móviles que junto con la incorporación del reconocimiento de imágenes e inteligencia artificial ayudará a determinar el número máximo de personas que puede acceder a una determinada zona y de esta forma controlar el acceso de nuevas personas a dicho espacio, cuando el aforo de este se haya completado.

Durante el desarrollo del aplicativo móvil se realizó la recopilación de información y análisis de requerimientos, y se diseñó cada una de las interfaces que conforman la aplicación y los diagramas de modelamiento correspondientes. Además, se llevó a cabo la implementación del proyecto en Android Studio que dio como resultado una aplicación que ofrece al usuario de manera muy intuitiva la posibilidad de controlar el aforo por medio de unas imágenes de entrada tomadas desde la cámara del dispositivo. Dicha imagen puede ser tomadas automática o manualmente y a partir de cada imagen se segmentan para el respectivo reconocimiento del rostro utilizando el módulo reconocimiento en cascada de OpenCv y de esta manera identificar y contar las personas en la imagen. Para validar el correcto funcionamiento de sistema se realizaron las respectivas pruebas, tomando como base diferentes escenarios se determinó que la capacidad de la aplicación es excepcional, si la cámara está en una buena posición (frontal) e iluminación se logra captar y contar, en la mayoría de los casos, todas las personas que están al margen de la cámara.

Palabras Clave: Aforo, Inteligencia Artificial, Reconocimiento de imágenes, Android, OpenCv.

Línea de Investigación: Tecnología y Sociedad.

INTRODUCCIÓN

En el último año debido a las nuevas disposiciones emitidas por el Gobierno Nacional para mitigar el contagio de COVID-19, los establecimientos comerciales con alto índice de aglomeración de personas como son los supermercados, bancos, restaurantes, etc. deben garantizar en todo momento mantener un número máximo de personas sugerido por las autoridades sanitarias, pero a pesar de todas las normativas no se está llevando un control del aforo adecuado (Rojas Campo, 2020). Esto se debe a que la mayoría de los establecimientos realizan el control por medio de un conteo manual que muchas veces es complicado e ineficiente. El conteo manual predispone terribles consecuencias y supone la utilización del recurso humano que bien podría enfocarse en otro tipo de tareas. Además, contratar personal para gestionar el flujo máximo de clientes, representa un costo mayor para los establecimientos comerciales.

Las diferentes herramientas de conteo de personas o los contadores manuales permiten controlar, con distinto grado de eficiencia, el número de personas entran y salen de un determinado lugar, sin embargo, este proceso es ineficiente. Por esta razón, la utilización de tecnologías como el reconocimiento de imágenes pueden mejorar el procedimiento de conteo y permitir a los dueños establecer un control de aforo en su establecimiento automáticamente. Así, el presente proyecto plantea el desarrollo de una aplicación para teléfonos móviles con sistema operativo Android que junto con la cámara del dispositivo y utilizando inteligencia artificial, sea capaz de identificar y contar personas en un determinado lugar de manera local y sin consumo excesivo de recursos. Al tratarse de una aplicación móvil permite a los establecimientos llevar un control económico, sin la necesidad de contratar nuevo personal o disponer de herramientas como cámaras o sistemas biométricos que en ocasiones suelen ser costosos. Para la elaboración de este proyecto se hizo un proceso de investigación y desarrollo de diferentes métodos de reconocimiento de imágenes y se seleccionó el más adecuado basándose en los algoritmos utilizados y la implementación, identificando así las funcionalidades y características más apropiadas para el sistema. Posteriormente se diseñaron las interfaces del usuario que permitiera visualizar en tiempo real el funcionamiento del aplicativo, además del modelamiento de cada uno de los diagramas de diseño. Por otra parte, se añadió el módulo de reconocimiento de imágenes (OpenCv) que por medio de algoritmos ya preestablecidos permiten identificar rostros a partir de una imagen. Además, se desarrolló la aplicación móvil nativa utilizando el IDE Android Studio, en donde se implementan todas las funcionalidades de Identificación y conteo de personas como también el registro de todos los datos realizados durante el día. Finalmente, se realizaron las pruebas pertinentes con la finalidad de analizar los resultados y determinar los rangos de funcionamiento del aplicativo y sus limitaciones.

1. PLANTEAMIENTO DE LA PROBLEMÁTICA

En el último año debido a las nuevas disposiciones emitidas por el gobierno nacional para mitigar el contagio de COVID-19, los establecimientos comerciales con alto índice de aglomeración de personas como son los supermercados, bancos, restaurantes, etc. deben garantizar en todo momento mantener un número máximo de personas sugerido por las autoridades sanitarias, pero a pesar de todas las normativas no se está llevando un control del aforo adecuado.(Rojas Campo, 2020) Esto se debe a que la mayoría de los establecimientos realizan el control por medio de un conteo manualmente que muchas veces es complicado e ineficiente. El conteo manual predispone terribles consecuencias y supone la utilización del recurso humano que bien podría enfocarse en otro tipo de tareas. Además, contratar personal para gestionar el flujo máximo de clientes, representa un costo mayor para los establecimientos comerciales.

Otra de las causas es la falta de cultura y mala adaptación de las personas a cumplir las medidas sanitarias propuestas por el gobierno nacional, dificultan de igual manera mantener el control de aforo, a pesar de mecanismos como: Avisos, barreras de acceso, sistemas de gestión de filas, etc. Las personas simplemente ingresan sin seguir ningunas de las normativas predispuestas, evitando lo protocolos de seguridad y poniendo en riesgo a otros clientes y empleados.

Por otra parte, las bajas ventas de los negocios hacen que los encargados tengan la necesidad de vender excesivamente y por esta razón verse obligados a dejar ingresar un mayor número de personas que las actualmente permitidas. Todo esto implica poner en riesgo el negocio hasta el punto de un posible cierre del local. Debido a la situación presentada, el proyecto plantea responder la siguiente pregunta: ¿De qué manera una aplicación móvil que utilice reconocimiento de imágenes ayudará a los establecimientos a llevar un control adecuado de aforo?

2. JUSTIFICACIÓN

Las diferentes herramientas de conteo de personas o los contadores manuales permiten controlar, con distinto grado de eficiencia, el número de personas entran y salen de un determinado lugar, sin embargo, este proceso es ineficiente. Por esta razón, la utilización de tecnologías como el reconocimiento de imágenes pueden mejorar el procedimiento de conteo y permitir a los dueños establecer un control de aforo en su establecimiento automáticamente. Así, el presente proyecto plantea el desarrollo de una aplicación para teléfonos móviles con sistema operativo Android que junto con la cámara del dispositivo y utilizando inteligencia artificial, sea capaz de identificar y contar personas en un determinado lugar de manera local y sin consumo excesivo de recursos.

Al tratarse de una aplicación móvil permite a los establecimientos llevar un control económico, sin la necesidad de contratar nuevo personal o disponer de herramientas como cámaras o sistemas biométricos que en ocasiones suelen ser costosos. La app podrá enfocar la cámara en los sitios de interés del usuario, con el fin de analizarlas por medio de unas redes neuronales artificiales entrenadas para la identificación y conteo de personas, y de esta manera controlar el aforo actual permitido en el establecimiento.

Instalar este sistema de control de aforo traería a los establecimientos comerciales una solución favorable que permitiría gestionar el flujo máximo de clientes, evitando así futuras aglomeraciones, garantizando el cumplimiento de las normativas y la seguridad de clientes, visitantes y empleados.

3. OBJETIVOS

3.1. Objetivo general

Desarrollar un sistema de control de aforo soportado en dispositivos móviles utilizando reconocimiento de imágenes e inteligencia artificial, con el fin de identificar y contar personas en un establecimiento comercial automáticamente.

3.2. Objetivos específicos

− Determinar los requerimientos de la aplicación, por medio de la investigación de tecnologías de reconocimiento de imágenes, identificando las funcionalidades y características más apropiadas para el sistema.

− Diseñar la arquitectura de la solución adecuada para la creación de un sistema de identificación de imágenes enfocada para dispositivos móviles por medio de diagramas de diseño y mockups.

− Implementar un módulo de reconocimiento de imágenes utilizando algoritmos de inteligencia artificial para identificar personas.

− Desarrollar la aplicación móvil nativa utilizando el IDE Android Studio, con el fin de identificar y contar personas en un establecimiento comercial automáticamente.

− Evaluar la aplicación en el entorno seleccionado por medio de escenarios de prueba validando su correcto funcionamiento.

4. ESTADO DEL ARTE

Tabla 1 Estado del Arte

Título	**Resumen**	**Aporte**	**Autores**
Reconocimiento de imágenes utilizando redes neuronales artificiales Madrid, España. Universidad Complutense de Madrid (Rogé et al., 2013)	En este proyecto se realizó un sistema de reconocimiento de imágenes, que por medio del teléfono móvil con sistema operativo Android se captura, procesa e identifica objetos mostrando la información de estos al usuario.	Este proyecto proporciona los métodos de reconocimiento de imágenes enfocado para dispositivos móviles, los cual son capaces de funcionar de manera local.	Pedro Pablo García García - José Antonio López Orozco
Age and gender classification using convolutional neural networks. Israel. The Open University of Israel. (Benkaddour et al., 2021)	En este proyecto se desarrolló un sistema de reconocimiento facial utilizando cámaras y redes neuronales convolucionales, en ella se segmentan las imágenes por capas y se logra identificar cada atributo para la clasificación automática de edad y género	El aporte del proyecto es cómo se implementa las Redes neuronales convolucionales (CNN) para la identificación facial.	Gil Levi - Tal Hassner
Diseño e integración en Android de un sistema de realidad aumentada y reconocimiento de imágenes para un sistema de domótica asistencial	Este proyecto se basa en una aplicación Android que utiliza realidad aumentada y reconocimiento de imágenes para manejar electrodomésticos por	El proyecto ayuda a entender los diferentes mecanismos de desarrollo de redes neuronales en un IDE como Android Studio.	Esteban Cobo Ceballos - Javier Fernández Muñoz

Madrid, España. Universidad Carlos III de Madrid. (Cobo Ceballos, 2013)	medio de voz en una casa domótica.		
Prototipo para el Control de Ingreso de Personal por Reconocimiento Facial Universidad Distrital Francisco José Caldas (Carin, A.A. & Sund, 2018)	En este proyecto se implementó un prototipo de control de acceso de personas mediante el uso de tecnología de reconocimiento facial basada en Deep learning utilizando la cámara web del computador	El aporte al proyecto es la amplitud del manejo de los conceptos de las tecnologías de reconocimiento facial basada en Deep learning.	Edinson Cáceres Parra
Propuesta de diseño de prototipo para el control de aforo y el distanciamiento social en Institución educativa de educación superior tecnológica de la ciudad de Barranquilla. Barranquilla, Colombia. Institución Educativa de Educación Superior Tecnológica.	Prototipo de mecanismo para el control de distanciamiento social y aforo en entidades educativas utilizando inteligencia artificial y la cámara proporcionada por el por computador.	El aporte al proyecto es que se implementan varias tecnologías de inteligencia artificial que pueden ser usadas en este proyecto, además de tratar una problemática similar.	Jose Alfredo Martinez Gandia-Javier Henriquez

(Martínez Gandia, 2020)			
Uso de redes neuronales para el reconocimiento de rostros en ambientes controlados. Universidad Distrital Francisco José de Caldas. (Montiel, 2015)	En este artículo se busca validar e implementar las redes neuronales en particular para identificación facial en sistemas biométricos y control de acceso.	Explican las etapas de: adquisición, extracción de parámetros faciales y la validación de estas en sistemas biométricos.	Holman Montiel Ariza - Fernando Martínez Santa - Diego Armando Giral Ramírez
Autenticación por reconocimiento facial para aplicaciones web, utilizando software libre. Bucaramanga, Colombia. Universidad Pontificia Bolivariana. (Web et al., 2012)	Es un proyecto que consta de un sistema de autenticación en aplicaciones web utilizando mecanismos como: Reconocimiento facial y sistemas biométricos, con el fin de evitar la suplantación de identidad.	El aporte al proyecto es la presentación de las técnicas usadas para la detección de caras en una imagen.	Carlos Guillermo Noguera Andrade
Sistema de reconocimiento facial con redes neuronales para la toma de asistencia en aulas de clase. Bucaramanga, Colombia. Universidad Autónoma de Bucaramanga	En este trabajo se realiza un sistema de reconocimiento facial para la toma de asistencia en las aulas de la universidad utilizando redes neuronales convolucionales con software y hardware de bajo costo.	Explican la arquitectura de un Sistema de reconocimiento facial SRF para lograr una mayor precisión y una definición más alta en la extracción de las capas en una imagen.	Miguel Eugenio Jurado García - Andrés Felipe Padilla Porras

(Jurado García & Padilla Porras, 2018)			
Representación de formas digitales para reconocimiento y clasificación de objetos. Universidad Autónoma de Bucaramanga. (Bergamini & Kamlofsky, 2015)	Este artículo, expone la implementación y evaluación de técnicas de análisis de imágenes orientadas al desarrollo de algoritmos para reconocimiento y clasificación de objetos en imágenes digitales.	El artículo presenta un algoritmo que puede ser útil para la identificación y comparación de imágenes.	María L. Bergamini, Jorge A. Kamlofsky
OpenStreetCam: reconocimiento automático de objetos en imágenes mediante machine learning. Madrid, España. UOC, Universidad a Distancia. (Jose Luis Villaluenga Morán, 2019)	Es un proyecto que utiliza imágenes aportadas por los usuarios para mapear cualquier localización geográfica que se presente. Manejando algoritmos basados en machine learning.	Refuerza los conceptos de Machine Learning dando a conocer los modelos y tipos de algoritmos comúnmente utilizado para su implementación en el reconocimiento de imágenes.	Jose Luis Villaluenga Morán
Aplicación de recomendaciones de moda basada en redes de aprendizaje profundo. Málaga, España. Universidad de Málaga.2019 (Ismael Pineda Palencia, 2019)	Este proyecto consta con una aplicación móvil en donde por medio de la cámara del dispositivo y utilizando redes neuronales profundas permiten la recomendación en tiempo real de artículos de moda.	Explican los modelos de red multicapa para el aprendizaje automático de las redes neuronales convolucionales.	Ismael Pineda Palencia

5. MARCO REFERENCIAL

5.1. Aforo

Aforo es el límite de capacidad que hay un determinado lugar, no obstante, en la actualidad se aplica como política sanitaria para mantener un número de personas permitidas localidades con espacios cerrados. Para asegurar el distanciamiento social en los establecimientos comerciales, se emplean avisos o carteles que muestran el límite predefinido de cantidad de individuos que pueden permanecer en dicho lugar. Este límite se establece según el área de la localidad y la distancia recomendada entre personas. (*¿Qué entendemos cuando hablamos del concepto aforo?*, 2020)

5.2. Covid-19

La enfermedad por coronavirus (COVID 19) es una enfermedad infecciosa causada por un coronavirus recientemente descubierto. La mayoría de las personas contagiadas experimentan síntomas de leves a moderados y se recuperan sin tratamiento especial, pero las mayores y con enfermedades cardiovasculares, diabetes, enfermedades respiratorias crónicas y cáncer tienen más probabilidades de desarrollar enfermedades graves.

El virus se transmite principalmente a través de gotitas de saliva o secreciones nasales cuando una persona infectada tose o estornuda, por lo que es importante que también practique la etiqueta respiratoria (Dennison Himmelfarb & Baptiste, 2020)

Figura 1 Estadística global del número de casos de COVID-19

Fuente: (Dong t al., 2020)

5.3. Inteligencia Artificial

La inteligencia artificial definida principalmente por Alan Turing como*: si una máquina puede actuar como un humano, entonces podremos decir que es inteligente.* Uno de los criterios para la inteligencia artificial es el llamado Test de Turing, la prueba se cumpliría si un humano interactuaba, sin saberlo, con una máquina y fuera incapaz de distinguir que esta no era una máquina (Teigens, 2020). En ese entonces se llevó a un amplio acuerdo entre los investigadores de las IA en que se requiere inteligencia para hacer lo siguiente:

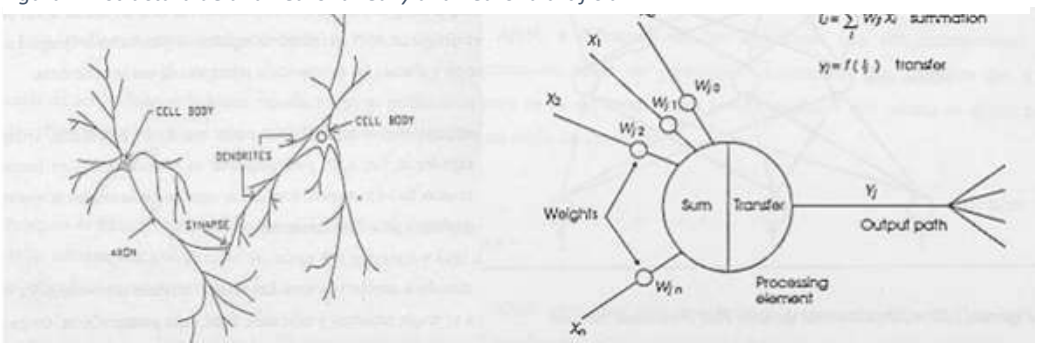
- Razonamiento
- Representar el conocimiento
- Aprendizaje
- Tratamiento de lenguajes naturales
- Integrar habilidades hacia objetivos comunes

Hoy en día, la inteligencia artificial es la ciencia que combina algoritmos planteados con el propósito de crear máquinas que presenten las mismas capacidades o similares a las del ser humano ya sea para tareas sencillas o tareas más complejas, es decir simulan la inteligencia humana por medio de sistemas informáticos. (García, 2012)

5.4. Redes neuronales Artificiales

En la rama de la inteligencia artificial, las redes neuronales artificiales (RNA) son un conjunto de algoritmos muy potentes con los que podemos modelar gran parte de comportamientos inteligentes. Se toma como base el sistema nervioso humano y su comportamiento, modelando así las neuronas y las conexiones en el cerebro para analizar y resolver problemas de la vida real.(Basogain Olabe, 2005)

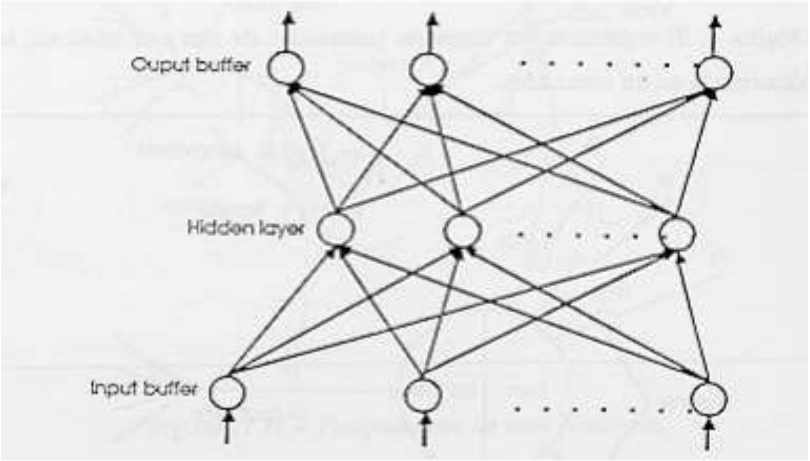
Figura 2: Estructura de una neurona real y una neurona artificial



Fuente: (Basogain Olabe, 2005)

Debido a su constitución y a sus fundamentos, las redes neuronales artificiales presentan un gran número de características semejantes a las del cerebro, como la capacidad de aprendizaje adaptivo, en donde cada una aprende a llevar a cabo ciertas tareas por medio de entrenamiento o con experiencia inicial.

Figura 3 Arquitectura de una red neuronal simple

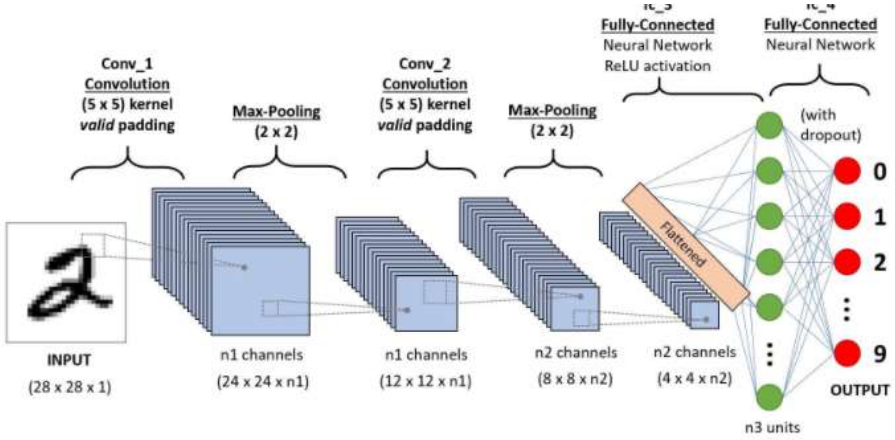


Fuente: (Basogain Olabe, 2005)

5.5. Rede Neuronal Convolucional (CNN)

Las redes neuronales basadas en operaciones de convolución o también denominadas CNN (Convolutional Neural Networks) es un algoritmo de aprendizaje profundo especializado en el procesamiento de imágenes, donde su función es tomar una imagen de entrada, asignar su importancia a varios objetos de la imagen y diferenciarla una de la otra (clasificación e identificación)(towardsdatascience, 2018).

Figura 4: Arquitectura de una red Neuronal Convolucional



Fuente: (towardsdatascience, 2018)

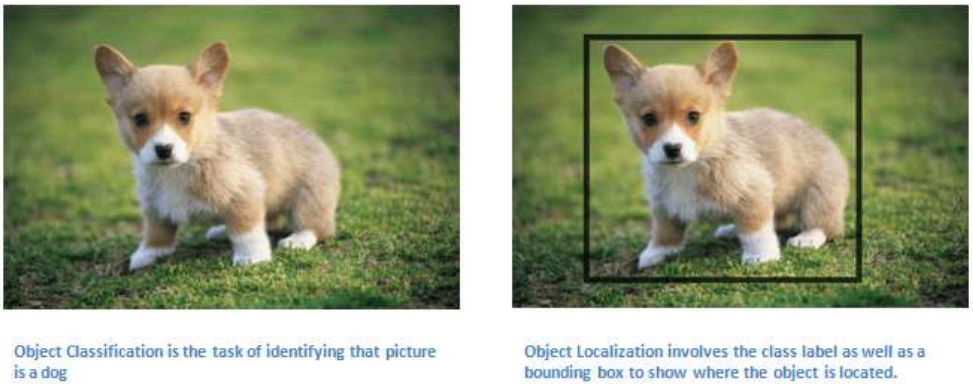
5.6. Reconocimiento de imágenes

El reconocimiento de imágenes computacional se basa principalmente en redes neuronales que aprenden a reconocer lo que representa cada imagen, identificando cada una y clasificándolas en una de varias clases distintas predefinidas, por lo tanto, define lo que se muestra en una imagen y distingue un objeto del otro(Rogé et al., 2013).

Está capacidad que tienen las máquinas para reconocer imágenes sirve como base para resolver diferentes problemas, que incluyen:

- Clasificación de imágenes con localización
- Detección de objetos
- Segmentación de objetos (semántica)
- Segmentación de instancias

Figura 5: Clasificación de imágenes por localización



Fuente: (Rogé et al., 2013)

5.7. Machine learning

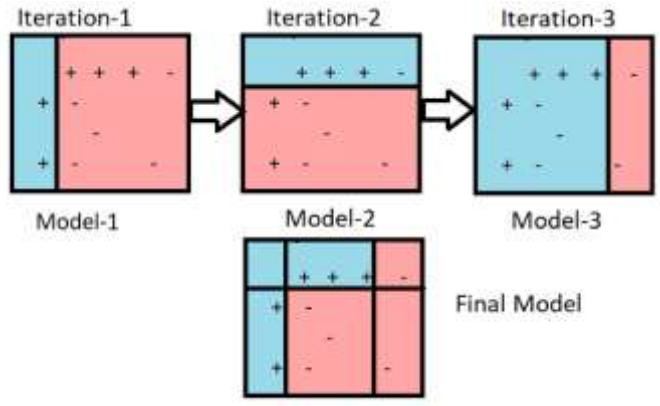
El aprendizaje automático (ML) es otra rama de la inteligencia artificial que permite a un sistema computacional aprender automáticamente a través de un conjunto de datos sin haber estado explícitamente programados. Es útil en cualquier escenario donde se requiera reconocer patrones, predecir comportamientos y tomar decisiones basados en millones de datos de entrada. De esta manera utiliza algoritmos para aprender dichos patrones de datos.

Los algoritmos de aprendizaje automático se clasifican en aprendizaje supervisado, no supervisado y de esfuerzo. En el aprendizaje supervisado los algoritmos usan datos que ya han sido cargados con anterioridad o con conocimiento previo para indicar como tendría que ser utilizada con la nueva información de entrada. Como su nombre lo indica se requiere que una persona esté supervisando para así proporcionar su respectiva retroalimentación. El aprendizaje no supervisado al contrario no maneja ningún dato con anterioridad, sino que tiene que encontrar la manera de clasificar dichos datos ellos mismos, buscando una estructura de esos datos de entrada para relacionarlos y organizarlos, por lo tanto, no requiere de una intervención humana. Por último, el aprendizaje de esfuerzo aprende de la experiencia. Es decir, se le "recompensa" al algoritmo cada vez que realiza un procedimiento correcto.(Rouhiainen, 2018)

5.8. Algoritmo AdaBoost

Los algoritmos boosting son aquellos métodos que buscan realizar una regla de precisión por la combinación de reglas débiles para formar una regla fuerte. AdaBoost (Adaptive boosting) es un diseño mejorado del planteamiento inicial del boosting, su objetivo es crear un clasificador fuerte con una precisión deseada cuya base sea la combinación de clasificadores débiles, esto lo realiza por medio de un entrenamiento realizando iteraciones de un conjunto de clasificadores débiles para la clasificación general de esta.(*Capítulo 3 Clasificadores Débiles-AdaBoost*, n.d.)

Figura 6 Funcionamiento algoritmo AdaBoost

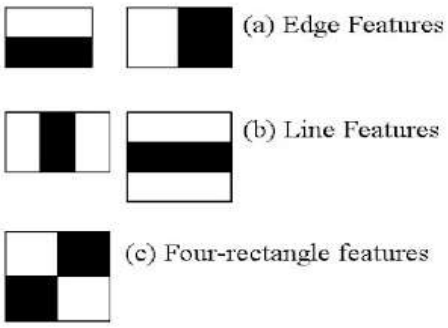


Fuente: (Rogé et al., 2013)

5.9. Características Haar

Las características Haar, son las características utilizadas en la visión artificial para la detección de objetos en imágenes digitales. Dicho clasificador permite extraer la características de una imagen o video, en donde cada una es un valor único obtenido al restar la suma de píxeles debajo del rectángulo blanco de la suma de píxeles debajo del rectángulo negro, como se muestra en la figura 7. Es decir, El cálculo de las características implica sumar las intensidades de píxeles en cada región y calcular las diferencias entre las sumas. (*OpenCV: Clasificador en cascada*, n.d.)

Figura 7 Características del clasificador haar

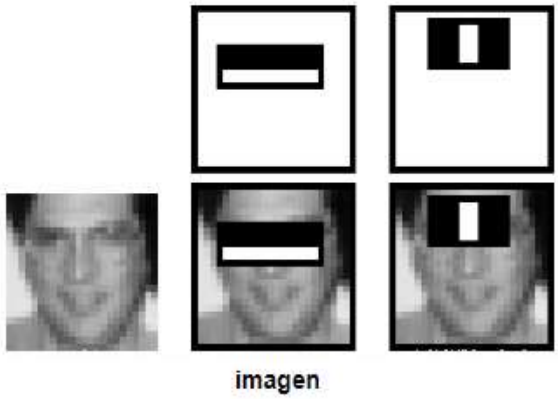


Fuente: (OpenCV: Clasificador en cascada, n.d.)

5.10. Clasificador Haar en cascada

La detección de objetos mediante clasificadores en cascada basados en características haar es un enfoque basado en aprendizaje automático propuesto por Paul Viola y Michael Jones, y tiene como base el algoritmo AdaBoost, realizando una versión actualizada de esta. El método consiste en entrenar el clasificador de cascada a partir de una gran cantidad de imágenes positivas y negativas, y por consiguiente extraer las características de él para luego detectar objetos o rostros como por ejemplo que la región de los ojos generalmente es más oscura que la nariz y las mejillas en otras imágenes.(*OpenCV: Clasificador en cascada*, n.d.)

Figura 8 Clasificador en cascada



Fuente: (OpenCV: Clasificador en cascada, n.d.)

5.11. Fases del método de Viola-Jones

El algoritmo Clasificador Haar en cascada planteado por Viola-Jones se realiza por medio de 3 etapas:

- **Imagen integral**

La imagen integral se utiliza para reducir el tiempo del cálculo de las operaciones de las características haar a diferentes escalas, este procedimiento consiste en tomar la imagen original y transformarla en una nueva imagen integral del mismo tamaño, en la que el resultado de la suma de los valores de todos los puntos situados por encima y a su izquierda en la imagen original contienen el valor del píxel seleccionado. (*Cascadas de Haar | de Aditya Mittal | Analytics Vidhya | Medio*, n.d.)

- **Extracción de las características**

El primer paso es extraer las características de la imagen por medio de funciones haar en cada una de estas funciones realizan los cálculo para obtener las regiones rectangulares adyacentes en una ubicación específica en una ventana de detección de la imagen. El cálculo implica sumar las intensidades de píxeles en cada región y calcular las diferencias entre las sumas. (*OpenCV: Clasificador en cascada*, n.d.)

- **Clasificación de las características**

  Para reducir los tiempos de procesamiento de las características se emplea el clasificador en cascada por medio del algoritmo de AdaBoost. El algoritmo selecciona las características con una tasa de error mínima y entrena a los clasificadores para que las utilicen. En este caso se utiliza un conjunto de clasificadores débiles para crear un clasificador fuerte y de esta manera el algoritmo de haar por medio de los filtros las pueda utilizar para detectar los rostros en un video o en imágenes de manera rápida. (*Cascadas de Haar | de Aditya Mittal | Analytics Vidhya | Medio*, n.d.)

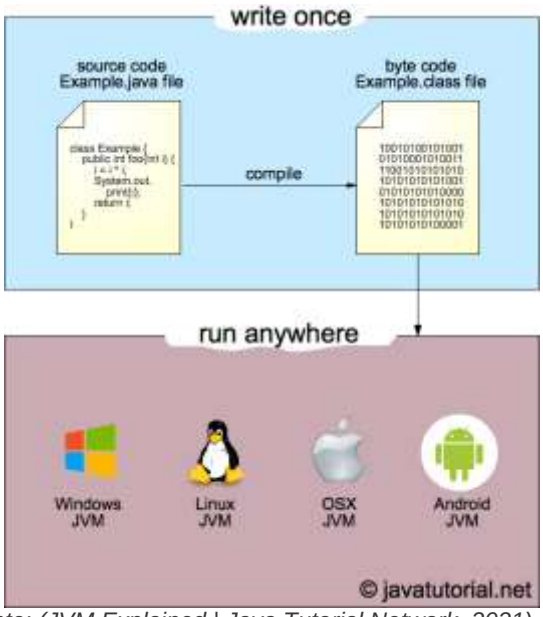
5.12. OpenCv

OpenCv es una biblioteca de código abierto de visión artificial desarrollada originalmente por Intel y disponible dede 1999. OpenCv se ha utilizado en infinidad de aplicaciones y desde diversos sistemas operativos como GNU/Linux, Windows y Mac OS X. Además, existe un desarrollo de interfaces para otros lenguajes de programación como Python, Ruby, Matlab, java, etc. Esta biblioteca Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estérea y visión robótica.(*Home - OpenCV*, n.d.)

5.13. JAVA

Originalmente desarrollado por la empresa Sun Microsystem en 1995 (2010 adquirida por Oracle), Java es un lenguaje de programación orientado a objetos, concurrente, de propósito general, rápido, seguro y fiable. Su arquitectura se basa en paquetes que permite agrupar las clases de un programa, y dentro de dichas clases agrupar los métodos, las variables, constantes, entre otros, que permite mantener una estructura jerárquica y ordenada. En la actualidad no hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado, se ve desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes. (Oracle, 2010).

Figura 9: Arquitectura JVM JAVA



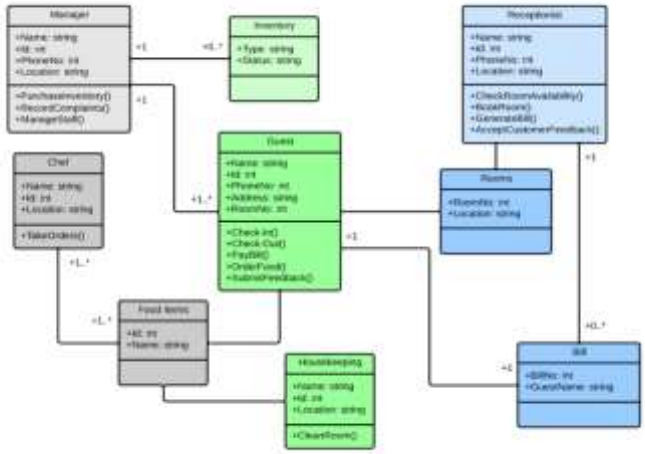
Fuente: (JVM Explained | Java Tutorial Network, 2021)

5.14. Programación orientada a objetos (POO)

La programación orientada a objetos es un paradigma de programación que se basa, en la utilización de objetos y clases, estas proporcionan conceptos y herramientas con las cuales se modela y representa el mundo real tan fielmente como sea posible. (Oracle, 2010)

La POO se divide en 3 pilares fundamentales: herencia, polimorfismo, encapsulación.

Figura 10: Diagrama de clases



Fuente: (Oracle, 2010)

5.15. Apps móviles

Una app es una aplicación de software que se ejecuta en un dispositivo móvil o tablet con el objetivo de ayudar a los usuarios en la realización de determinadas acciones. Estas facilitan el proceso de compra de los usuarios, fomentan la interconectividad y mejora la experiencia de usuario.(Cadenas, 2019)

5.15.1. Tipos de Apps

- Nativas: Son aquellas que se diseñan y desarrollan en un sistema operativo específico, empleando también un lenguaje de programación específico. Este tipo de aplicaciones son estables y permiten obtener el máximo rendimiento del dispositivo.(Cadenas, 2019)

- Híbridas: Estas al igual que las nativas, se pueden instalar en dispositivos, pero se ejecutan desde navegadores web y son desarrolladas en lenguajes de programación HTML5. Aunque las apps no son tan rápidas o fiables como las nativas, su proceso de desarrollo es más rápido. (Cadenas, 2019)

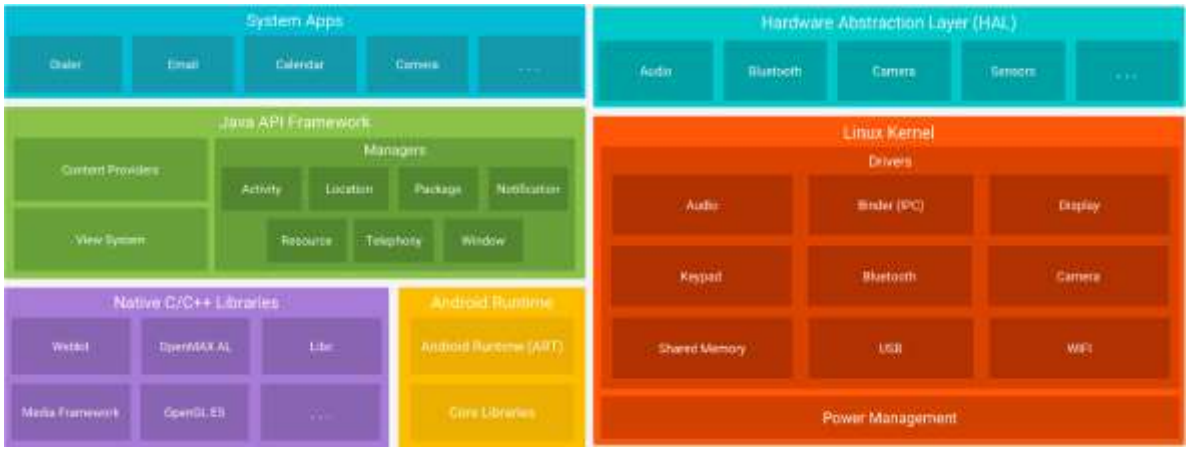
Figura 11: Aplicaciones nativas vs Aplicaciones Híbridas

Fuente: (Cadenas, 2019)

5.16. Android

Android es un sistema operativo de código abierto basado en Linux, diseñado principalmente para dispositivos móviles como smartphones y tabletas. Su funcionamiento permite manipular los dispositivos de manera intuitiva y admite una gran cantidad de aplicaciones, además es uno de los SO más utilizados en la actualidad. La Arquitectura de Android se basa en el kernel de Linux donde s u principal uso es la gestión de la memoria, los procesos, las operaciones de red y otros servicios del sistema operativo (Méndez, 2015).

Figura 12: Arquitectura Android



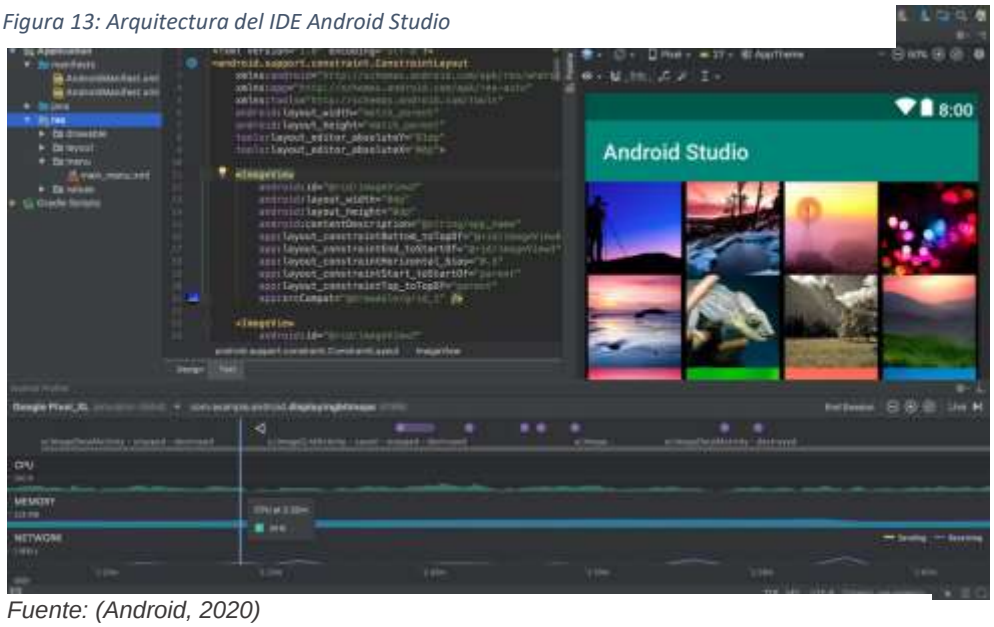
Fuente: (Méndez, 2015)

5.17. IDE Android Studio

Android Studio es entorno de desarrollo integrado (IDE) utilizado en el sistema operativo Android para el desarrollo de aplicaciones móviles. El IDE posee un sistema de compilación flexible basado en Gradle que permite un emulador rápido y cargado de funciones. Además, ofrece la posibilidad de escribir código en Kotlin y Java, si bien gracias al Android NDK se incluye la opción de dar soporte a lenguajes como C o C++.

La Integración con GitHub y plantillas de código también ayuda a compilar funciones de apps comunes y importar código de muestra.(Android, 2020)

Figura 13: Arquitectura del IDE Android Studio

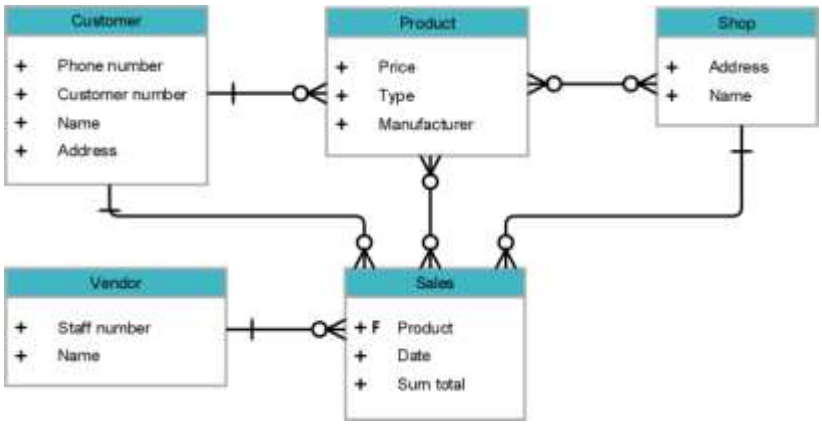


Fuente: (Android, 2020)

5.18. Base de datos

Una base de datos es una colección organizada y estructurada de datos que está almacenada electrónicamente en un sistema computacional, generalmente controlada por un sistema de gestión de base de datos (DBMS) que le permite acceder a los datos, manipularlos, generar informes y representar dichos datos.(*¿Qué es la base de datos NoSQL? Guía completa de la base de datos NoSQL*, 2019).

Figura 14: Modelo Entidad Relación



Fuente: (¿Qué es la base de datos NoSQL? Guía completa de la base de datos NoSQL, 2019)

5.19. Sqlite

SQLite Es un sistema de gestión de base de datos relacional pequeño, rápido, autónomo, de alta confiabilidad y con todas las funciones. Además, es el motor de base de datos más utilizado del mundo, está integrado en todos los teléfonos móviles y la mayoría de las computadoras, y viene incluido dentro de innumerables aplicaciones que la gente usa todos los días.(sqlite, 2018)

5.20. Android JetPack

Son un conjunto de librerías que permiten a los desarrolladores buscar una alternativa sencilla para seguir las prácticas recomendadas, reducir las líneas de código y escribir código que funcione de manera coherente en los dispositivos para que pueda enforcarse en el código que les interesa.(*Android Jetpack | Desarrolladores de Android | Android Developers*, n.d.)

Figura 15 Arquitecura Android Jet-Pack

*Fuente: (*Android Jetpack | Desarrolladores de Android | Android Developers*, n.d.)*

5.20.1. Room Database

Room es una librería de persistencia de base de datos basada sobre SQLITE, permite beneficiarse con la posibilidad de conservar datos de manera local. (*Cómo guardar contenido en una base de datos local con Room*, n.d.) los 3 componentes principales de Room:

- Base de datos: Sirve como punto de acceso principal para la conexión subyacente a los datos persistentes y relacionales de la aplicación. La clase anotada con @Database debe cumplir con las siguientes condiciones:
    - Ser una clase abstracta que extiende RoomDatabase.

    - Incluir la lista de entidades asociadas con la base de datos dentro de la anotación.

    - Contener un método abstracto que tenga 0 argumentos y muestre la clase anotada con @Dao En el entorno de ejecución, puedes adquirir una instancia de Database llamando a Room.databaseBuilder() o Room.inMemoryDatabaseBuilder().

- Entidad: Representa una tabla dentro de la base de datos.

- DAO: Contiene los métodos utilizados para acceder a la base de datos.

6. DESARROLLO DE LA SOLUCIÓN

6.1. Análisis de requerimientos

En esta sección se describirá según con la información recopilada en las entrevistas y encuestas la especificación de los requerimientos de la aplicación, definiendo los requisitos y las funcionalidades a ofrecer.

6.1.1. Investigación de tecnologías

Se llevo a cabo un proceso de investigación de las diferentes tecnologías de reconocimiento de imágenes más utilizadas. En la siguiente tabla se pueden observar las tecnologías encontradas y sus diferentes características. Se seleccionó el más adecuado basándose en las pruebas de cada uno en los repositorios de código investigados, teniendo en cuenta la implementación móvil, tiempos de procesamiento y los algoritmos utilizados.

Tabla 2 Tabla de comparación de tecnologías

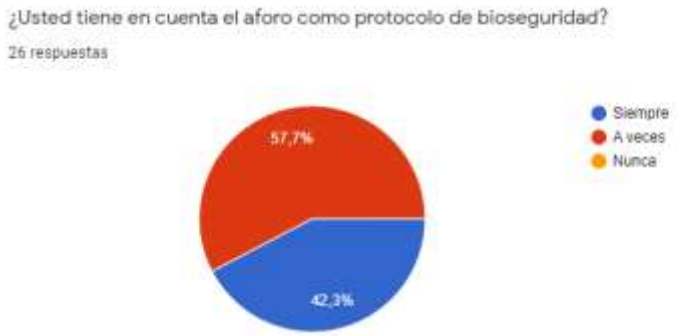
Tecnología	Referencia	Lenguaje de programación	Descripción	Características de reconocimiento	Implementación móvil
Opencv		C++, Python, Java y MATLAB	Es una biblioteca de software de visión abierta y software de aprendizaje automático.	Cuenta con un conjunto completo de algoritmos de visión artificial y de aprendizajes automáticos tanto clásicos como avanzados. Estos algoritmos se pueden usar para detectar y reconocer rostros, identificar objetos	Sí
YOLO		Python	es un sistema para detección de objetos en tiempo real, el cual hace uso de una única red neuronal convolucional para detectar objetos en imágenes.	Utiliza deep learning y CNN para detectar objetos.	Sí
TensorFlow		Python	Es una librería de software open source tiene la capacidad de crear una CNN para clasificación e identificación de imágenes.	Arquitectura multicapa, estas capas son seguidas por capas conectadas que conducen a un clasificador softmax y por las regresiones lineales se puede identificar la imagen.	Sí
KERAS		Python	es una biblioteca de Redes Neuronales de Código Abierto que está diseñada para posibilitar la experimentación en más o menos poco tiempo con redes de Aprendizaje	Genera modelos de deep learning en teléfonos inteligentes.	Sí

Fuente: Propia

6.1.2. Encuestas

Se elaboro 10 preguntas con el objetivo de recopilar información referente al aforo que está llevando como medida de control sanitario en los establecimientos públicos como: bancos, restaurantes, tiendas, hoteles, etc. A continuación, se hace una descripción de los resultados obtenidos tomando en cuenta las dos preguntas más importantes:

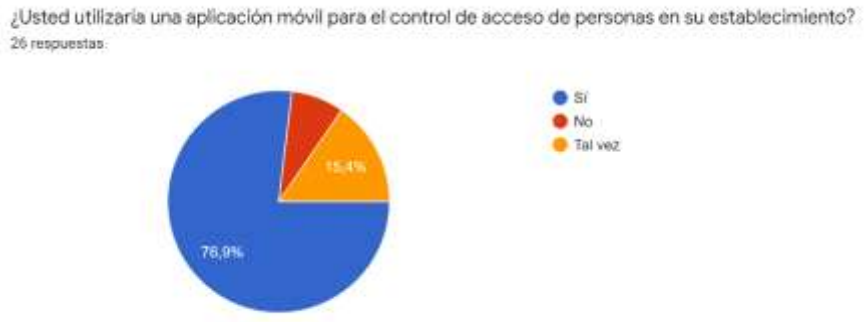
Figura 16 Encuesta pregunta #3



Fuente: Propia

De las 26 personas encuestadas el 57,7% a veces tienen en cuenta el aforo como protocolo de bioseguridad. Mientras tanto el 42,3% siempre considera el aforo como protocolo.

Figura 17 Encuesta pregunta #10



Fuente: Propia

De acuerdo con la anterior pregunta el 76,9% de las 26 personas encuestadas afirman utilizar la aplicación móvil para el control de acceso de personas. El 15,4% estima tal vez hacer uso del aplicativo y el 7.7% asegura no manejar este tipo de tecnología.

6.1.3. Entrevistas

Para la metodología cualitativa se seleccionaron tres personas para el desarrollo de las entrevistas. Dos de ellas trabajan en establecimientos comerciales; y uno es dueño de un local comercial. En la siguiente tabla muestra los datos obtenidos:

Tabla 3 Descripción entrevistas

Hechos	**Aspectos negativos**	**Aspectos positivos**	**Atajos**	**Dilemas**	**Ideas**
Se tiene un leve conocimiento acerca del aforo.	Dejan ingresar más personas para el beneficio propio.	Encuentran buena opción para utilizar una aplicación móvil para el control de aforo.	Las mismas personas que atienden realizan el control de aforo.	Dicen mantener la seguridad de los clientes en todo momento, pero dejan ingresar más personas que las que están permitidas.	Utilizan rutas de acceso.
Mantienen en todo momento la seguridad de los clientes.	Gastan dinero adicional en contratar personal para controlar el acceso.	La mayoría de los establecimientos se responsabilizan en la seguridad de los clientes.			
Hay al menos una persona encargada de llevar el control de aforo.	Las personas no mantienen la distancia requerida y hacen caso omiso a los avisos y carteles.	Conocen la existencia de las normativas predispuestas por el gobierno nacional.			
	Un entrevistado manifiesta que le es difícil manejar herramientas tecnológicas.				

Fuente: Propia

Teniendo en cuenta la información anterior se evidencia que hay un leve conocimiento del aforo, pero por aspectos económicos resuelven atender más cantidad de personas de las que están permitidas. Los dueños y trabajadores de

estos locales son quienes vigilan por su propia cuenta el ingreso de personas, lo que afecta de antemano el riesgo de contagio. Por último, ellos afirman hacer uso de tecnologías de reconocimiento móviles para controlar el aforo teniendo en cuenta los costos que generan contratar adicional para la revisión del conteo.

A pesar del desconocimiento del uso de estas tecnologías se considera que son herramientas que aportan de manera significativa la posibilidad de trabajar con seguridad, reduciendo gastos y cuidando la salud del personal a cargo y los clientes.

6.1.4. Requerimientos del sistema

Con respecto a la información recopilada en la investigación de las tecnologías de reconocimiento de imágenes, las encuestas y las entrevistas se plantearon los requerimientos que van a hacer parte de la funcionalidad del sistema:

Requerimientos no funcionales

- El sistema debe tener una interfaz agradable al usuario, intuitiva y fácil de utilizar.
- La aplicación debe correr sobre el sistema operativo Android.
- Debe poder ser accedido desde cualquier dispositivo Android con nivel api 23 o superior.
- La aplicación no consumirá más de 1 GB de RAM.
- La aplicación no podrá ocupar más de 1 GB de espacio en disco.
- La aplicación móvil debe poder funcionar sin la necesidad de ningún tipo de conexión a internet. Definirla bien
- El sistema debe funcionar con la cámara estándar del dispositivo.

Requerimientos funcionales

- El sistema debe darle la opción al usuario de registrarse e ingresarse.
- El sistema debe contar con un apartado de instrucciones de uso.
- El sistema debe tomar las imágenes correspondientes.
    - Se debe dar la opción de tomar las fotos donde el usuario desee identificar y contar las personas.
    - Se debe mostrar en pantalla la foto que se ha tomado.
- El sistema debe identificar a las personas.
    - Se debe extraer las características de las fotos tomadas por la cámara.

- Se debe identificar a cada una de las personas presentes.
- La identificación se representará por medio de un recuadro que se posicionará en la parte superior de la persona.
- El sistema no realizará ninguna caracterización de las personas identificadas (género, edad, entre otras)

- El sistema debe tomar las fotos automáticamente.
  - Se debe mostrar un apartado donde el usuario indique el tiempo que necesita que se tome la foto: 5s, 1m, 5m, 10m, respectivamente.
  - tomará las fotos automáticamente y las guardará en el respectivo registro, teniendo en cuenta el tiempo ingresado por el usuario

- El sistema debe realizar un conteo de las personas.
  - Se debe mostrar un apartado donde el usuario indique el número de personas límite permitidas en su establecimiento.
  - Se debe mostrar el número de personas que están actualmente.

- El sistema debe validar que se cumple la cantidad de personas permitidas.
  - Se validará si se sobrepasa el límite establecido.

- El sistema debe registrar datos.
  - Se registran los datos generales del local.
  - El sistema guardará internamente los datos en el dispositivo.
  - Se registran las imágenes tomadas durante el día.
  - Se registran los datos tomados por el sistema de conteo que ocurrieron durante las sesiones diarias.

- El sistema debe mostrar los datos guardados
  - El sistema debe mostrar una lista de los días en donde se realizó el control de aforo.
  - El sistema debe mostrar cada uno de los registros tomados durante ese día en específico, indicando: Foto tomada, Nombre, Fecha, Hora y cantidad máxima de personas.
  - El sistema debe tener la opción de filtrar por medio de un botón los registros que sobrepasaron el límite de personas y de esta manera mostrarlos.

6.2. Diseño

Este capítulo está centrado en el diseño de la aplicación, realizando así las interfaces que va a contener la aplicación y los diagramas de modelamiento. El detalle de los elementos presentados en esta fase se plasma en el documento de arquitectura el cual se puede visualizar en el **Anexo *B* Documento de arquitectura**.

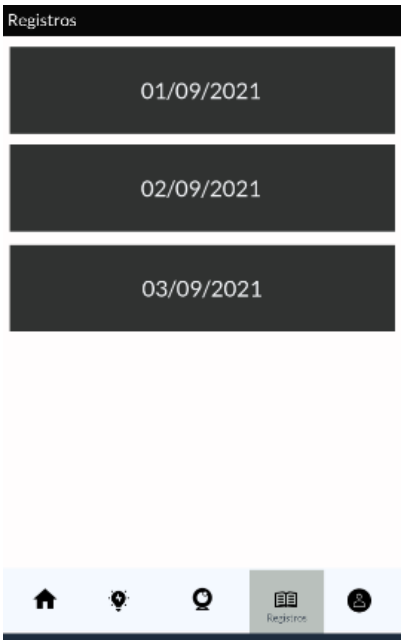
6.2.1. Interfaces

Se han diseñado las interfaces graficas muy sencillas en la plataforma online MarverlApp, allí se crean cada una de las pantallas que contiene el sistema y se emplea el respectivo mokcup. Cada una de las interfaces, además de ser intuitivas y de fácil manejo para el usuario, también nos permiten visualizar correctamente todos los pasos que se han ido dando en el desarrollo de la aplicación, entre las más importantes tenemos:

Figura 18 Interfaz cámara

Fuente: Propia

Figura 19 Interfaz Registro



Fuente: Propia

Figura 20 Interfaz de detalle

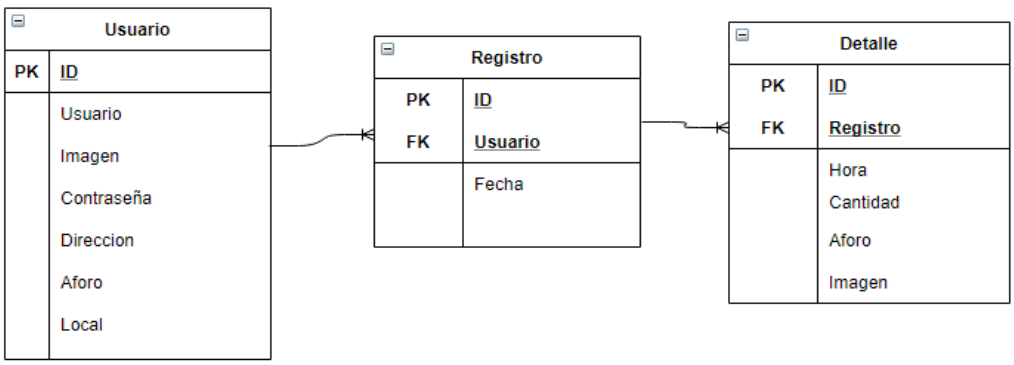
Fuente: Propia

6.2.2. Modelo relacional

En este modelo se plantearon cada una de las entidades que componen el sistema junto con sus respectivas relaciones.

- Usuario: Guarda los atributos correspondientes al usuario: Nombre de usuario, foto del local, dirección, capacidad de aforo y contraseña.
- Registro: Guardan cada una de las fechas por día en que se realizó un control de aforo.
- Detalle: Guardan los datos sacados de la toma de la imagen: La hora en que se tomó la foto, La cantidad de personas identificadas, la capacidad que se está manejando y la respectiva imagen tomada.

Figura 21 Modelo Relacional

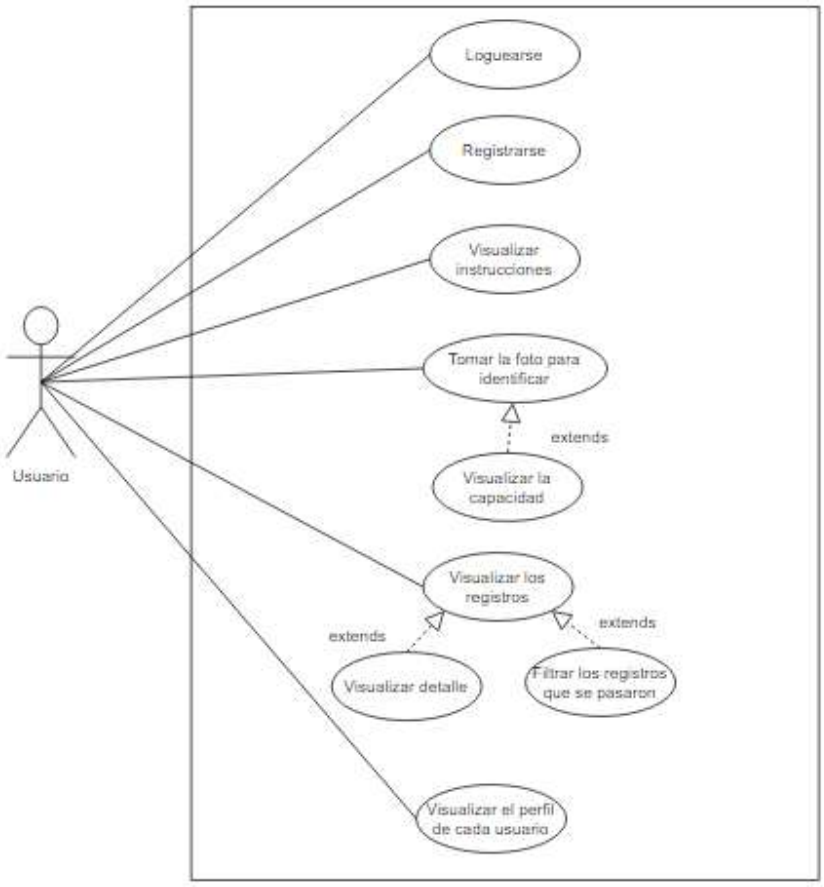


Fuente: Propia

6.2.3. Diagrama de casos de uso

En este diagrama se plantearon todos los actores con sus respectivos casos de usos los cuales afectan o intervienen en el sistema. Se puede ver en el recuadro los casos de uso que contiene el usuario con respecto a las funcionalidades del sistema.

Figura 22 Diagrama de casos de uso



Fuente: Propia

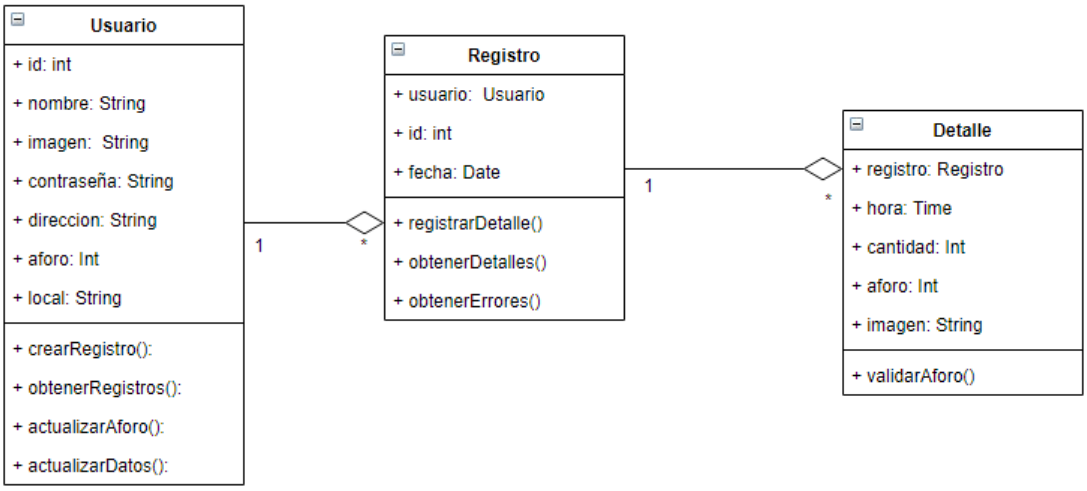
Diagrama de clases

Este diagrama nos muestra cada una de las clases que componen nuestro sistema y sus relaciones, entre ellas tenemos:

- Usuario: Son las personas que van a ingresar y utilizar el sistema.
- Registro: Son los registros que se realizaron durante el día.

Detalle: Son los detalles asociados a cada uno de los registros ocurridos durante el día, además filtra los datos con base al aforo.

Figura 23 Diagrama de clases



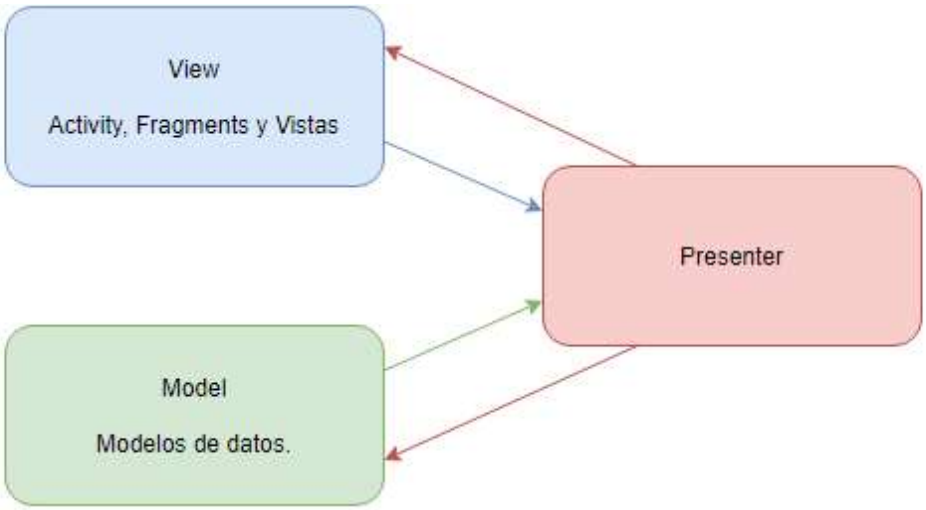
Fuente: Propia

6.2.4. Diagrama de arquitectura MVP

La arquitectura planteada para este proyecto es el patrón de diseño MVP (Modelo Vista Presentador).

- Vista: Dentro del entorno se elaborará una interfaz visual interactiva (Vista), que contará con diferentes clases y vistas como por ejemplo:
    - Vistas: Login, Registro, Inicio, Instrucciones, cámara, listado de registros, detalle y perfil
    - Activitys y frament relacionados a estas vistas
- Presenter: Es la parte que recibe los datos y los valida, originados por la vista y así como el encargado del flujo de datos por medio de clases que implementan la vista y el modelo.
- Modelo: Es la información que se muestra y genera (Lógica del negocio), esta se obtiene de la comunicación con la base de datos (ROOM), esta contiene:
    - Entidades: usuario, registro y detalle
    - Dao: UsuarioDao, RegistroDao y DetalleDao
    - La base de datos

Figura 24 Diagrama de Arquitectura MVP

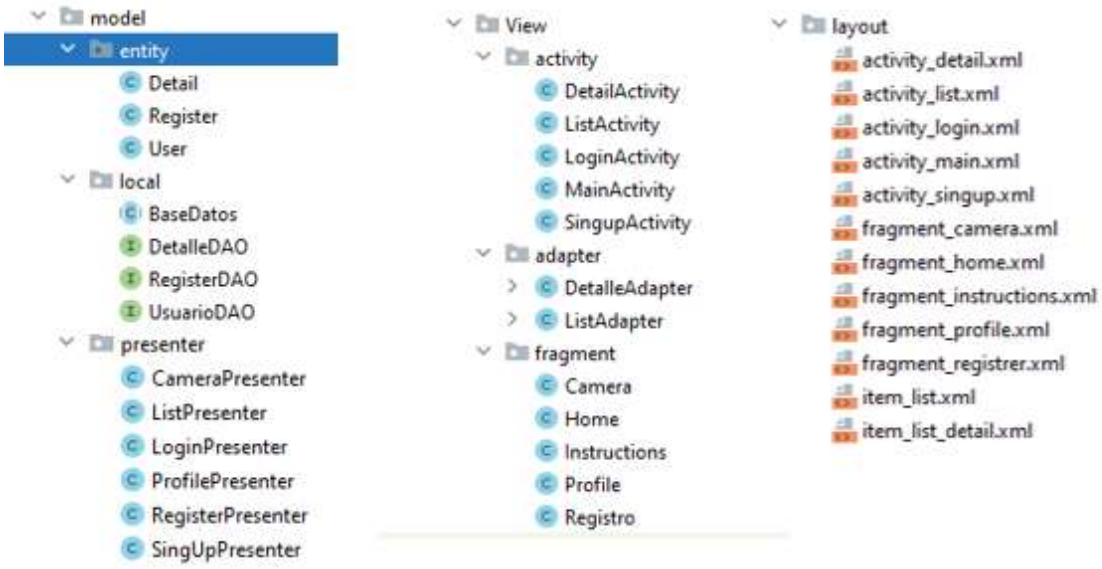


Fuente: Propia

6.3. Implementación

En esta sección se describirá todo lo relacionado al entorno de desarrollo para la realización del proyecto, como la arquitectura de desarrollo a utilizar, las librerías externas utilizadas y modelamiento de datos, mostrando así su uso en el código del proyecto.

Figura 25 Estructura del proyecto en Android Studio

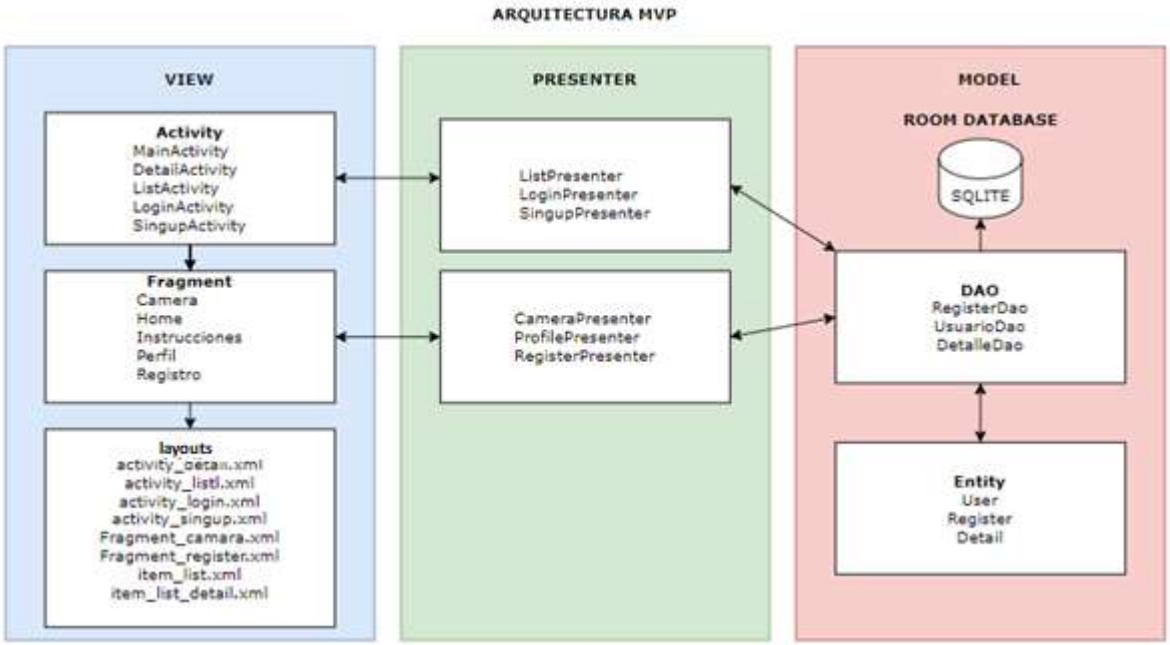


Fuente: Propia

6.3.1. Arquitectura de desarrollo

Como se menciona anteriormente se trabajó el desarrollo del proyecto bajo una arquitectura MVP, de esta manera se dividirán las clases que correspondan a cada carpeta y se crearán los presentadores que conectarán las vistas con el modelo. En la siguiente imagen se muestra toda arquitectura de desarrollo y su respectivo flujo de datos:

Figura 26 Arquitectura de desarrollo MVP



Fuente: Propia

6.3.2. Base de datos

Se utilizó como sistema de gestión de base de datos la biblioteca de Room, esta brinda alta capacidad de abstracción para Sqlite y permite trabajar base de datos de persistencia de manera sencilla y sin complicaciones.

6.3.2.1. Implementación de dependencias

- Se agregan las dependencias de los artefactos en el archivo build.gradle de la aplicación

```
implementation "androidx.room:room-runtime:2.2.6"
annotationProcessor "androidx.room:room-compiler:2.2.6"
```

6.3.2.2. Entidades

Tomando en cuenta en los diagramas de entidad relación y de clases se representaron cada una de las clases de la aplicación en la base de datos, en donde

se obtienen y se configuran los valores que corresponden a columnas de tabla dentro de la base de datos. Además se usan cada una de las anotaciones que conforman la entidad (@Enity, @ ForeignKey, @ColumnInfo) para que Room las reconozca como una función en Room.

- Se crean cada una de las notaciones: para que Room las reconozca.
- Se crea la entidad con los campos requeridos.
- Se crean el constructor de la entidad y sus getters and setters.

```
@Entity(tableName = "Detail", foreignKeys = {@ForeignKey(entity =
Register.class,
        parentColumns = {"id"},
        childColumns = {"register_id"})
})
public class Detail implements Serializable{
    @ColumnInfo
    @PrimaryKey(autoGenerate = true)
    public int id;
    @ColumnInfo(name ="Hour")
    private String hour;
    @ColumnInfo(name ="url_imagen")
    private String image;
    @ColumnInfo(name ="Aforo")
    private int aforo;
    @ColumnInfo(name ="Cantidad")
    private int amount;
    @ColumnInfo
    public int register_id;
```

6.3.2.3. Dao

Los DAO contienen los métodos utilizados para acceder a la base de datos, de esta manera la aplicación usa cada DAO para obtener las entidades y guardar los cambios realizados en esas entidades en la base de datos. Al igual que en las entidades se usan las anotaciones @Query, @ Insert para definir los Dao y las respectivas consultas dentro de Room.

```
@Dao
public interface DetalleDAO {

    @Query("select * from Detail where register_id = :id")
    List<Detail> obtenerDetalleporRegistro(int id);
    @Query("select * from Detail inner join Register on
Detail.register_id = Detail.id ")
    List<Detail> obtenerDetalleByInner();
    @Insert
    void agregar(Detail detail);
}
```

6.3.2.4. Base de datos (Room)

Contiene el titular de la base de datos y sirve como punto de acceso principal para la conexión subyacente a la persistencia de datos y las relacionales dentro de la aplicación.

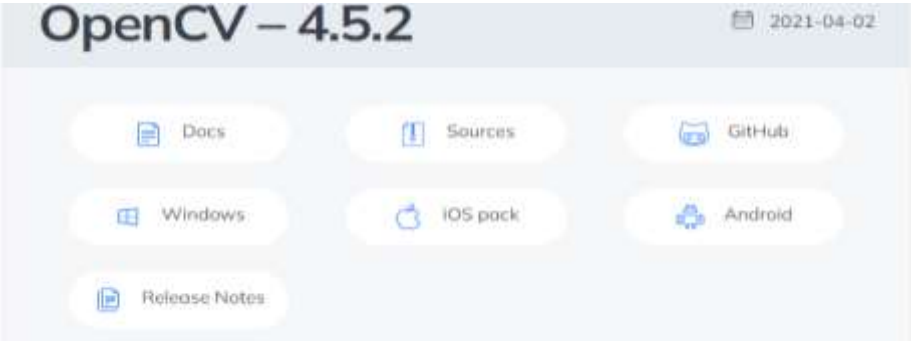
```
@Database(entities = {User.class ,Register.class, Detail.class},
version = 1)
public abstract class BaseDatos extends RoomDatabase {

    public abstract UsuarioDAO usuarioDAO();
    public abstract DetalleDAO detalleDAO();
    public abstract RegisterDAO registerDAO();
    private static BaseDatos instancia= null;

    public static BaseDatos getInstancia(Context context){
        if(instancia ==null){
            instancia = Room.databaseBuilder(context,
BaseDatos.class, "aforo.db").allowMainThreadQueries().build();
        }
        return instancia;
```

6.3.3. Opencv

Cómo módulo de reconocimiento de imágenes se agregó la librería Opencv, ya que su fácil implementación nos permite utilizar gran variedad de algoritmos de inteligencia artificial, entre ellos el clasificador en casada que permite reconocer los rostros de las personas en la aplicación. La versión de la librería de OpenCV para Android utilizada ha sido: OpenCV-4.5.2. A continuación se mostrará la implementación de la librería y su utilización en las clases más importantes de la aplicación:

6.3.3.1. Implementación de la librería

- Descargar la librería por la página oficial de OpenCv en este caso la 4.5.2.



Fuente: Propia

- En el archivo de build.gradle y luego en default.config agregar:

```
applicationId "opencv.org"
```

- Agregamos la siguiente dependencia y Luego en sincronizar:

```
testImplementation 'junit:junit:4.13.2'
```

- Se copia el sdk del módulo que descargamos anteriormente.

Fuente: Propia

- Se pega el sdk dentro del archivo de nuestra aplicación.

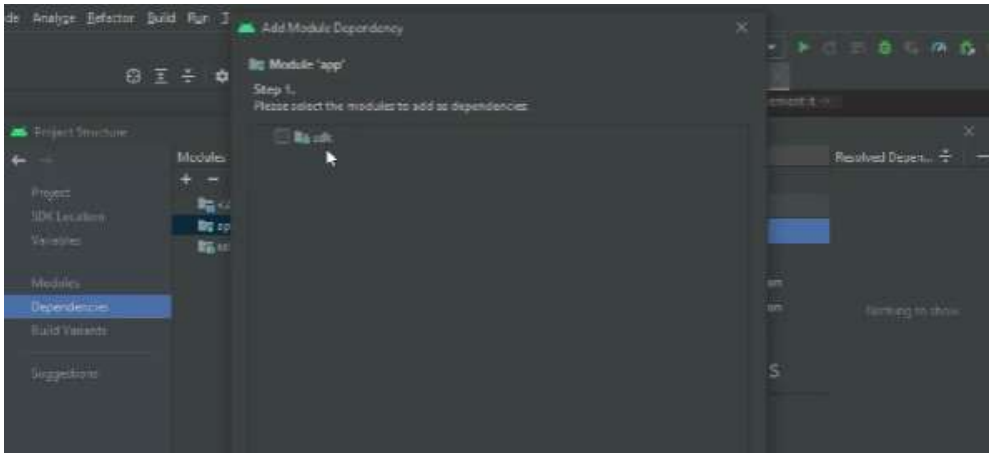
.gradle	17/11/2021 11:07 p. m.	Carpeta de archivos	
.idea	18/11/2021 12:35 a. m.	Carpeta de archivos	
app	29/10/2021 07:36 a. m.	Carpeta de archivos	
gradle	20/09/2021 08:26 p. m.	Carpeta de archivos	
sdk	20/10/2021 10:46 a. m.	Carpeta de archivos	
.gitignore	20/09/2021 08:26 p. m.	Documento de te...	1 KB
build.gradle	30/09/2021 03:22 p. m.	Archivo GRADLE	1 KB
gradle.properties	20/09/2021 08:26 p. m.	Archivo PROPERTI...	2 KB
gradlew	20/09/2021 08:26 p. m.	Archivo	6 KB
gradlew	20/09/2021 08:26 p. m.	Archivo por lotes ...	3 KB
local.properties	20/09/2021 08:26 p. m.	Archivo PROPERTI...	1 KB
settings.gradle	19/10/2021 06:00 p. m.	Archivo GRADLE	1 KB

Fuente: Propia

- Se incluye el sdk dentro del archivo de settings.gradle y sincronizamos.

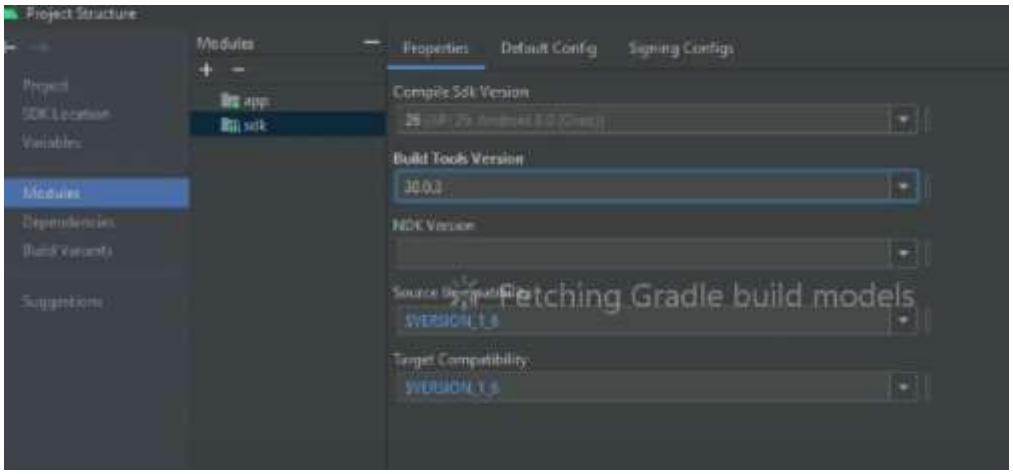
```
include ':sdk'
```

- Ahora se va a la opción “project structure” y añadimos el módulo de la dependencia y oprimimos en “apply” y “ok”



Fuente: Propia

- Para finalizar se cambia la versión de “Builds Tools Version” a 30.0.3.



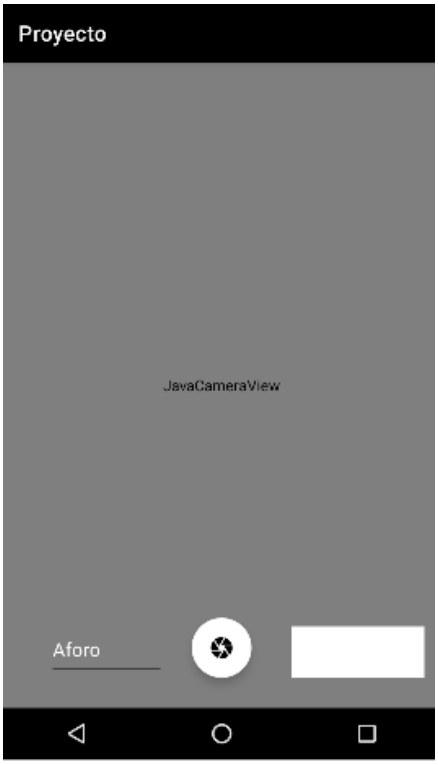
Fuente: Propia

- Para verificar que la librería se ha cargado correctamente se realiza el siguiente código:

```
public void onManagerConnected(int status) throws IOException {
    super.onManagerConnected(status);
    switch (status){
        case BaseLoaderCallback.SUCCESS:
            Log.i(TAG, "OpenCV loaded successfully");
            javaCameraView.enableView();
            initializeOpenCVDependencies();
            break;
        default:
            super.onManagerConnected(status);
            break;
    }

}
```

6.3.3.2. Camera fragment OpenCv

Figura 27 Interfaz de desarrollo Camera Fragment



Fuente: Propia

- Implementa el listener para obtener los frames de la cámara.

```
public class Camera extends Fragment implements
CameraBridgeViewBase.CvCameraViewListener2 {
```

- Crea la instancia de la cámara cargada en la vista y la inicializa.

```
private void initializeCamera(JavaCameraView javaCameraView, int
activeCamera){
    javaCameraView.setCameraIndex(activeCamera);
    javaCameraView.setVisibility(CameraBridgeViewBase.VISIBLE);
    javaCameraView.setCvCameraViewListener(this);
```

- Inicia la carga asíncrona de la librería.

```
public void onResume() {
    super.onResume();
    if (!OpenCVLoader.initDebug()) {
        OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION,
getContext(), mLoaderCallback);
    } else {
        try {

mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS)
;
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

- Obtiene el frame del cámara recibido en la aplicación para su respectivo procesamiento.

```
public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame
inputFrame) {
    mRgba = inputFrame.rgba();
    return mRgba;
}
```

6.3.4. Detección de rostros

Esta funcionalidad es la encargada de reconocer los rostros presentes en la imagen, para ello se va a utilizar el clasificador en cascada que es provista por la librería OpenCv a través de un archivo Xml, en este caso se va a utilizar la: haarcascade_frontalface_alt2.xml.

6.3.4.1. Clasificador en cascada

- Agrega la dependencia del archivo xml que contiene la función de reconocimiento de caras.

- Se agrega el xml en un archivo para utilizarlo posteriormente.

- Se obtiene la cadena de la ruta del archivo.

```
private void initializeOpenCVDependencies() throws IOException {

    InputStream is =
getResources().openRawResource(R.raw.haarcascade_frontalface_alt2);
    File cascadeDir = getActivity().getDir("cascade", MODE_PRIVATE);
    cascFile = newFile(cascadeDir,"haarcascade_frontalface_alt2.xml");
    FileOutputStream os = new FileOutputStream(cascFile);

    cascadeClassifier =
newCascadeClassifier(cascFile.getAbsolutePath());

}
```

6.3.4.2. Obtener Imagen

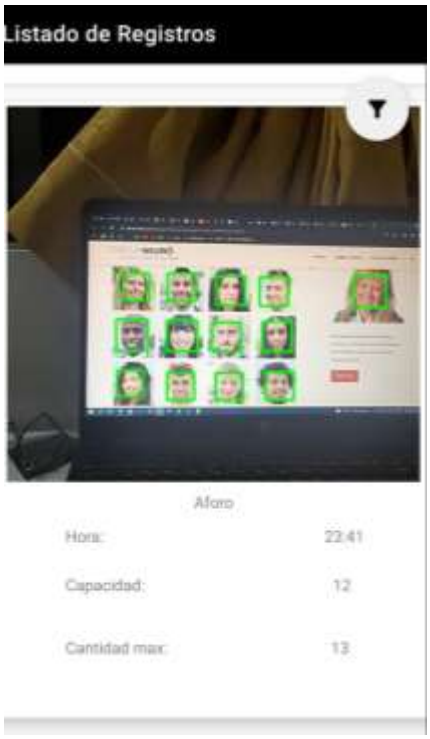
- Convierte el frame entrante en un mapa de bits (Bitmap).
- Modifica la imagen en escala de grises.
- Convierte el objeto a tipo Mat para realizar la respectiva identificación.

```
private void saveImage(Mat subImg) {
    Bitmap bmp = null;
    try{
        bmp = Bitmap.createBitmap(subImg.cols(),
subImg.rows(),Bitmap.Config.ARGB_8888);
        grayscaleImage = new Mat(bmp.getHeight(), bmp.getWidth(),
CvType.CV_8UC4);
        Utils.matToBitmap(drawRect(subImg), bmp);
```

6.3.4.3. Identificación de personas y conteo

La funcionalidad de identificación y conteo de personas permite al usuario tomar las fotos y que automáticamente identifique las personas, guarde las imágenes y cuente las personas por cada imagen obtenida.

Figura 28 Interfaz de detalle del aplicativo



Fuente: Propia

Funcionalidad en código:

- Modifica el frame a la escala de grises.

- Se ejecuta la función detectMultiScale para detectar las caras. Se necesitan 3 argumentos:
  - Imagen de entrada (Rgba) y el objeto (faceDetection).
  - Se especifica cuanto se reduce el tamaño de la imagen con cada escala (1.1).
  - Se especifica el número mínimo de cuadros (7 y 2).

- Se guarda en el objeto “amount” el tamaño de la lista dado por el array **MatOfRect** para realizar el respectivo conteo de las personas identificadas dentro de la imagen.

- Se itera la lista para dibujar el recuadro en las detecciones teniendo en cuenta las coordenadas de las caras en la imagen.

```
public Mat drawRect(Mat rgba) {
    Imgproc.cvtColor(rgba, grayscaleImage, Imgproc.COLOR_RGBA2RGB);
    if (mDetectorType == 0) {
        if (cascadeClassifier != null) {
            MatOfRect faceDetection = new MatOfRect();
            cascadeClassifier.detectMultiScale(rgba, faceDetection,
1.1, 7, 2);
            amount = faceDetection.toList().size();
            for (Rect rect : faceDetection.toArray()) {

                Imgproc.rectangle(rgba, new Point(rect.x, rect.y),

                  new Point(rect.x + rect.width, rect.y +
rect.height),
                        FACE_RECT_COLOR, 3, 2);
            }
        }
    }
    return rgba;
}
```

6.3.5. Almacenamiento de las imágenes

- Se crea el formato de cómo se va a guardar la imagen, en este caso por la fecha y hora de creación.

- Se crea el nombre del archivo y la ruta donde se va a crear el archivo.

- Guarda la ruta de la imagen(path) para enviarla a la base de datos y de esta manera llamarla desde otra funcionalidad.

```
 Log.i(TAG,"onTouch event");
@SuppressLint("SimpleDateFormat") SimpleDateFormat sdf = new
SimpleDateFormat("yyyy-MM-dd_HH-mm-ss");
String currentDateandTime = sdf.format(new Date());
String fileName =  File.separator+currentDateandTime + ".jpg";
path= Environment.getExternalStorageDirectory().getPath()+
"/images_"+fileName;
File sd = new
File(Environment.getExternalStorageDirectory(),"/images_");
```

- Si la carpeta ya está creada guarda la imagen, sino crea la carpeta.

- Se comprime la imagen a png y se determina la calidad con que se va a guardar la imagen.

```
boolean success = sd.exists();
if(!success){
    success = sd.mkdir();
}
if(success){
    File dest = new File(sd,fileName);
    try {
        out = new FileOutputStream(dest);
        bmp.compress(Bitmap.CompressFormat.PNG, 50, out);

    } catch (Exception e) {
        e.printStackTrace();
        Log.d(TAG, e.getMessage());
    }
```

6.3.6. Funcionalidad de toma automática de las fotos

La funcionalidad de toma automática permite al usuario realizar la identificación y el conteo de las personas automáticamente, tomando en cuenta los diferentes tiempos disponibles que maneja la opción de la cámara (5s, 1m, 5m y 10m).

Figura 29 Interfaz de la cámara automática

Fuente: Propia

Funcionamiento en código:

- Se crea el array que contiene los diferentes tempos que se van a utilizar para la toma automática de imágenes.

- Se agrega el array en el spinner.

```
String [] tempo = {"Ninguna","5s","1m","5m","10m"};
ArrayAdapter<String> adapter = new
ArrayAdapter<String>(getContext(),android.R.layout.simple_spinner_item
,tempo);
spinner.setAdapter(adapter);
```

- Se guarda la opción seleccionada.

- Se hace un switch con las opciones del array.

```
selected = spinner.getSelectedItem().toString();
Toast.makeText(getContext(),"Modo automatico activado, tempo:
"+spinner.getSelectedItem().toString(), Toast.LENGTH_SHORT).show();
switch (selected){
```

- Se crea en cada una de las opciones un handler que permite ejecutar una función cada cierto tiempo, en este caso ejecuta “cargardatos()” que es el método encargado de realizar todas las funciones de la toma de las fotos, las identificación y el almacenamiento de estas.

    - Se utiliza la opción handler.removeCallbacksAndMessages(null) para pausar la función.

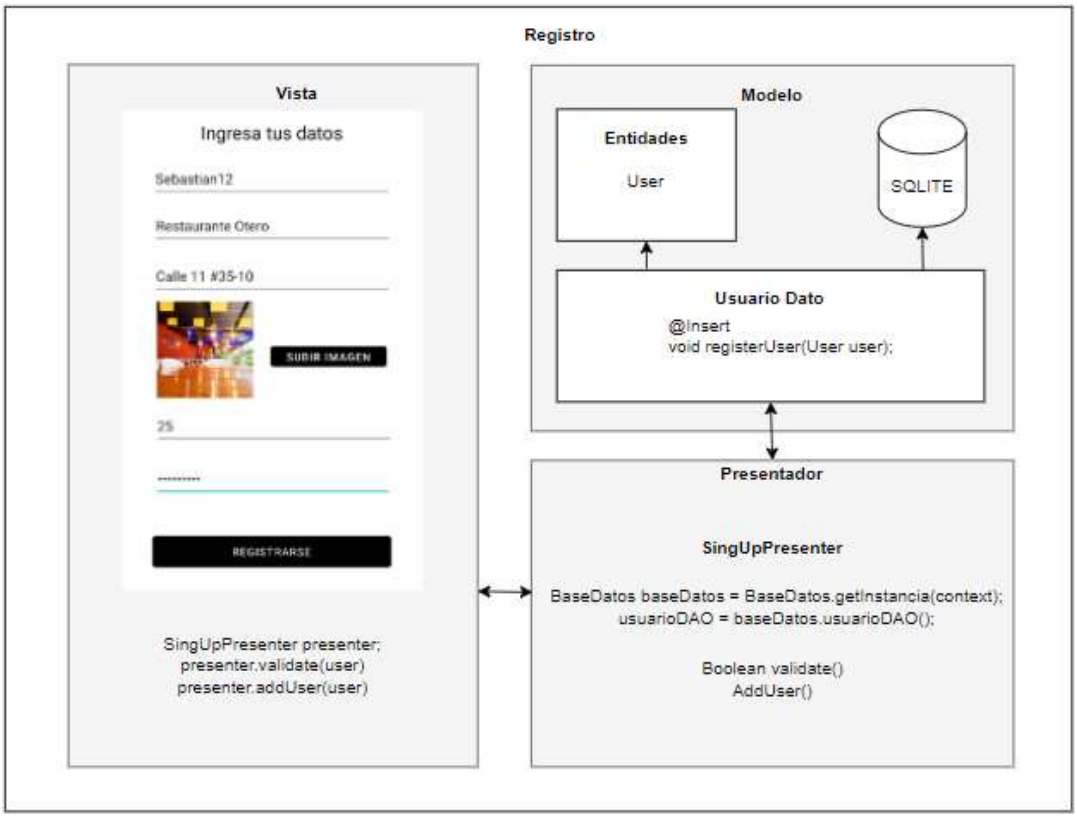
    - Se agrega el delay correspondiente al tiempo establecido.

```
case "Ninguna":
    handler.removeCallbacksAndMessages(null);
    break;

case "5s":
    handler.removeCallbacksAndMessages(null);
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {

            Log.e("Tiempo: ","5s");
            cargarDatos();
            handler.postDelayed(this,5000);
        }
    },5000);
```

6.3.7. Registro de usuario

Se realizó la funcionalidad de registro del usuario, en la que por medio de un formulario se agregan la información general y una imagen que se añade desde la galería. Por medio de la siguiente figura se muestra el flujo de datos del registro:

Figura 30 Flujo de datos Registro de usuario



Fuente: Propia

6.3.8. Login

La funcionalidad de login permite al usuario loguearse al sistema teniendo en cuenta el nombre del usuario y la contraseña, manteniendo así su sesión activa dentro de la aplicación. Esta contiene solo una validación sencilla que consulta si el usuario y la contraseña se encuentran dentro de la base de datos y permite el acceso al sistema.

Figura 31 Interfaz de login.

Fuente: Propia

6.3.9. Lista de los registros diarios

Se realizó la lista de registros diarios que contiene cada uno de los detalles(fotos) tomados en el apartado de la cámara. Está funcionalidad al momento de tomar la foto crea un registro automáticamente teniendo en cuenta el día en que se tomó y de esta manera lo muestra en la lista. Si el registro del día no existe lo crea y si existe agrega los detalles correspondientes a ese día.

Funcionamiento en el código:

- Se crea un Adapter que contiene el recycler view que permite listar los cada ítems del registro
- Se llama el Adapter dentro del fragmento del registro y se emplean cada uno de los métodos para crear la lista de registros con cada uno de los ítems dentro del recyclerView.

```
adapter = new DetalleAdapter(listadoDetalle);
rvDetail.setAdapter(adapter);
rvDetail.setLayoutManager(new
LinearLayoutManager(getApplicationContext()));
rvDetail.setHasFixedSize(true);
```

Figura 32 Interfaz de listado de registros por día

Fuente: Propia

6.3.10. Listado de detalles

El listado de detalles viene a partir del registro del día anteriormente mencionado, este permite obtener todos los datos relacionados con la toma de la foto y se muestran por medio de una lista. Cada uno de los ítems contiene: La imagen con las respectivas personas identificadas, La hora en la que se tomó la foto, el aforo y el número de personas identificadas en la foto.

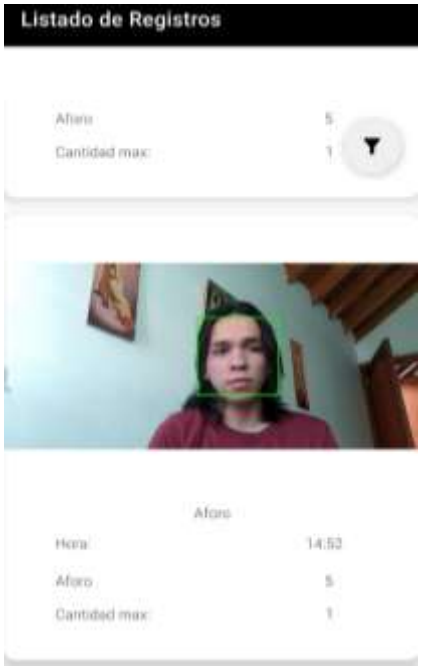
Funcionamiento en código:

- Obtiene los datos del detalle en la base de datos.
- Obtiene la dirección de la imagen en un string.
- Posteriormente la convierte la imagen que está relacionada a esa dirección en un mapa de bits y carga al ImageView de la vista.

```
Detail detail = (Detail) getIntent(). getSerializableExtra("detail");
tvHora.setText(String.valueOf(detail.getHour()));
tvAforo.setText(String.valueOf(detail.getAforo()));
tvamount.setText(String.valueOf(detail.getAmount()));
String path = detail.getImage();
BitmapFactory.Options options = new BitmapFactory.Options();
options.inPreferredConfig = Bitmap.Config.ARGB_8888;
```

```
bmpInput = BitmapFactory.decodeFile(path, options);
ivAforo.setImageBitmap(bmpInput);
```

Figura 33 Interfaz de listado de detalles



Fuente: Propia

6.3.11. Filtro

La funcionalidad de filtro permite filtrar los datos por aquellos que se sobrepasaron del límite, es decir aquellos en el que el Aforo sea menor a la cantidad máxima de personas identificadas. De esta manera al oprimir en el botón de filtrado (1) se listen solamente aquellos los datos en los que la “cantidad max” sea mayor a la del aforo.

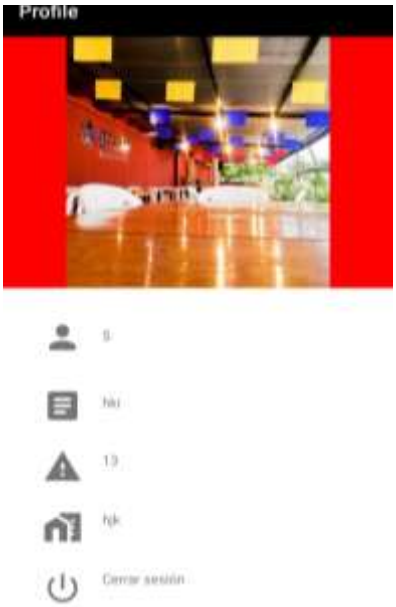
Figura 34 Interfaz de listado de detalle + filtrado

Fuente: Propia

6.3.12. Perfil

El perfil carga los datos creados en el registro, que contiene, Nombre de usuario, Dirección, Aforo y la imagen subida anteriormente desde la galería. Además, permite al usuario por medio de un botón cerrar la sesión.

Figura 35 Interfaz funcionamiento perfil



Fuente: Propia

6.4. Pruebas

En este capítulo se definen las pruebas de la aplicación con el objetivo de verificar la funcionalidad de los distintos elementos que componen el sistema. Se realizan en escenarios controlados verificando que la respuesta de las operaciones coincidía con los requisitos establecidos en la fase de análisis. Para entrar en detalle de cada uno de los escenarios y sus respectivas pruebas ir al documento de escenarios de prueba que se encuentra en el **Anexo *C* Escenario de pruebas**.

Tabla 4 Prueba funcionalidad Registro

Prueba 1	Registro del usuario
Descripción	Registro del usuario al sistema
Estado inicial	La aplicación debe estar instalada
Estado final	El sistema guarda la información del usuario y la imagen ingresada
Procedimiento	Se van a crear 5 usuarios con diferentes datos para verificar que se agreguen correctamente en la base de datos.
Resultado Obtenido	Éxito

Fuente: Propia

Tabla 5 Prueba funcionalidad Login

Prueba 2	Login
Descripción	Login para que el usuario ingrese al sistema
Estado inicial	Haberse registrado con anterioridad
Estado final	Sesión iniciada
Procedimiento	Se iniciará sesión con los 5 usuarios anteriormente creados
Resultado Obtenido	Éxito

Fuente: Propia

Tabla 6 Prueba funcionalidad perfil

Prueba 3	Perfil
Descripción	Apartado de para visualizar los datos personales del usuario y la función de cerrar sesión

Estado inicial	Debe estar logueado en sistema
Estado final	Se visualizan todos los datos del usuario en el perfil.
Procedimiento	Se verifica que en cada uno de los usuarios creados se muestren todos sus datos y que la imagen cargue correctamente. Además de que la opción de cerrar sesión funcione bien.
Resultado Obtenido	Éxito

Fuente: Propia

Tabla 7 Prueba Identificación manual

Prueba 4	Identificación y conteo de personas manual
Descripción	Realiza la toma de las fotos, identificación de las personas y obteniendo un conteo de estas manualmente.
Estado inicial	• Cámara activada • Aforo diligenciado • Tempo “ninguno”
Estado final	• Visualización de la foto tomada • Identifica las personas en la imagen • Guardar la imagen en el dispositivo • Guarda la ruta de la foto y los datos de día, hora, aforo y capacidad en la base de datos • Se visualiza todos la lista de detalles de cada uno de los registros tomados
Procedimiento	Se van realizan 5 tomas manuales en diferentes lugares, perfiles, y con aforos diferentes para comprobar que se realice la identificación y conteo de manera eficiente.
Resultado Obtenido	Todas las imágenes se guardan correctamente, pero la identificación de las personas tiene un porcentaje de 80% de exactitud ya que existen factores como la luz y la posición del rostro que impiden el reconocimiento de esta.

Fuente: Propia

Tabla 8 Prueba funcionalidad toma de fotos automática

Prueba 5	Identificación y conteo de personas automáticamente
Descripción	Realiza la toma de las fotos, identificación de las personas y el conteo de las personas de manera automática, tomando en cuenta el tiempo establecido por el usuario (5s, 1m, 5m o 10m).
Estado inicial	• Cámara activada. • Aforo diligenciado. • Tiempo seleccionado.
Estado final	• Toma de la foto cada cierto tiempo(tempo). • Guardar las imágenes en el dispositivo y las lista en el detalle. • Identifica las personas en cada una de las imágenes.
Procedimiento	Se va a realizar la identificación y el conteo de manera automática en un entorno seleccionado durante tres horas, teniendo en cuenta que el tiempo base es de 5m.
Resultado Obtenido	Dado que el dispositivo se ubicó en un lugar amplio y con buena iluminación en la mayoría de los registros en los que hay una persona de por medio las logra identificar. Y por consiguiente guardar los datos extraídos correctamente.

Fuente: Propia

Tabla 9 Prueba funcionalidad registros por día

Prueba 6	Guardar registros del día
Descripción	Lista de registros diarios que contienen todos los detalles (registros de tomas) realizados durante el día
Estado inicial	Haber realizado alguna toma de control de aforo, ya sea manual o automática.
Estado final	Carguen los datos de cada item y se listen cada uno.
Procedimiento	Tomando en cuenta los registros realizados en pruebas anteriores, se verifican que se cree el registro diario correspondiente.
Resultado Obtenido	Éxito.

Fuente: Propia

Tabla 10 Prueba funcionalidad detalle

Prueba 7	Guardar los detalles de las fotos
Descripción	registros y visualización del detalle de los datos extraídos al momento de tomar la foto
Estado inicial	Haber realizado alguna toma de control de aforo, ya sea manual o automática.
Estado final	Se guardan los datos y se visualizan en el detalle de cada registro.
Procedimiento	Tomando en cuenta los registros realizados en pruebas anteriores, se verifican que se guarden cada uno de los datos en los detalles y se visualicen en un listado.
Resultado Obtenido	Éxito.

Fuente: Propia

CONCLUSIONES

Al hacer la recopilación de las tecnologías de reconocimiento se evidencia que el uso de las librerías de reconocimiento de imágenes como OpenCv es la opción más viable, por su baja complejidad computacional y poca utilización de recursos de la imagen. Además, su fácil implementación en dispositivos móviles, permiten utilizar todas las funcionalidades y la gran variedad de algoritmos de reconocimiento que contiene está librería.

Por otra parte, Las pruebas demostraron un porcentaje de eficacia bastante alto al momento de identificar las personas en las imágenes, sin embargo no siempre se logran captar en las fotos teniendo en cuenta que existen múltiples condiciones que afectan los algoritmos de reconocimiento junto con la iluminación, así como la posición y la pose de la cara, afecta al momento de reconocer a las personas, importante destacar que el sistema de reconocimiento fue construido para hacer la identificación en solo la parte frontal del rostro, por lo tanto dependiendo de la posición de la persona el porcentaje identificación varia un poco.

Sin embargo, la capacidad de la aplicación es excepcional, si la cámara está en una buena posición (frontal) y iluminación se logra captar y contar, en la mayoría de los casos, todas las personas que están al margen de la cámara. A pesar de los pocos recursos que maneja un dispositivo móvil para ejecutar algoritmos de reconocimiento, los tiempos de respuesta son rápidos y se logran obtener buen resultado al momento de realizar el control de aforo y llevar un registro de estos.

Por último, cabe aclarar que el trabajar con una arquitectura limpia MVP permitió actualizar las funcionalidades en la aplicación de una manera más organizada y rápida, sin afectar todas las demás configuraciones o acciones que se desarrollaron durante en el proyecto.

RECOMENDACIONES O TRABAJOS FUTUROS

El Desarrollo de este proyecto como se evidenció, busca el control del aforo utilizando algoritmos de inteligencia artificial para identificar y contar personas en un determinado lugar automáticamente, esto se realiza tomando fotos manualmente o automáticamente y de esta manera por medio de identificadores en cascada captan las personas en las imágenes y realiza el respectivo conteo, por lo tanto, se espera realizar como trabajo futuro enfocar la aplicación de control de aforo no solo en reconocer imágenes si no también hacer todo el procedimiento de identificación y conteo en tiempo real (Video), ya sea seguir utilizando los algoritmos de OpenCv o incluso optar por la realidad aumentada, así siendo un sistema mucho más robusto, tecnológico y con mayor precisión que el prototipo actual.

Adicionalmente, extrapolar todo el sistema de reconocimiento a una plataforma online, el cual, por medio de un servidor cada cámara o dispositivo sea un punto de vista y donde cada una de las tomas de identificación se interpolaran, se llega a obtener un panorama más amplio y de mayor alcance. Además, todos estos datos podrían ser analizados por medio de un portal web de administración para los tiempos de ejecución, tipos de captura, y otro portal para centralizar todo los videos y fotos tomadas durante el día lo que podría incrementar todo el potencial del proyecto.

.

REFERENCIAS BILIOGRÁFICAS

¿Qué entendemos cuando hablamos del concepto aforo? (2020). https://www.geovictoria.com/cl/que-es-aforo/

¿Qué es la base de datos NoSQL? Guía completa de la base de datos NoSQL. (2019). https://www.educba.com/what-is-nosql-database/

Android Jetpack | Desarrolladores de Android | Android Developers. (n.d.). Retrieved November 23, 2021, from https://developer.android.com/jetpack?gclid=EAIaIQobChMI0NLO4a6v9AIV0cqGCh0a9gaCEAAYASAAEgJgnfD_BwE&gclsrc=aw.ds

Android. (2020). *Arquitectura de la plataforma | Desarrolladores de Android*. https://developer.android.com/guide/platform?hl=es-419

Basogain Olabe, X. (2005). Redes Neuronales Artificiales Y Sus Aplicaciones. In *Medicina Intensiva* (Vol. 29, Issue 1). http://linkinghub.elsevier.com/retrieve/pii/S021056910574198X

Benkaddour, M. K., Lahlali, S., & Trabelsi, M. (2021). Human Age and Gender Classification using Convolutional Neural Network. *2020 2nd International Workshop on Human-Centric Smart Environments for Health and Well-Being, IHSH 2020*, 215–220. https://doi.org/10.1109/IHSH51661.2021.9378708

Bergamini, M. L., & Kamlofsky, J. A. (2015). Representación de formas digitales para reconocimiento y clasificación de objetos. *Revista Colombiana de Computación*, *16*(1), 28–47. https://doi.org/10.29375/25392115.2492

Cadenas, R. (2019). *¿Que necesito? ¿Web Apps, App Nativa o App Híbrida? - GSoft*. https://www.gsoft.es/articulos/que-necesito-web-apps-app-nativa-o-app-hibrida/

Capítulo 3 Clasificadores Débiles-AdaBoost. (n.d.).

Carin, A.A. & Sund, R. . (2018). *Prototipo Para El Control De Ingreso De Personal Por Reconocimiento Facial* (Issue 1) [UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS]. https://repository.udistrital.edu.co/bitstream/handle/11349/13637/C%E1ceresParraEdinson2018.pdf?sequence=1

Cascadas de Haar, explicado. Una breve introducción a Haar… | de Aditya Mittal | Analytics Vidhya | Medio. (n.d.). Retrieved December 1, 2021, from https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d

Cobo Ceballos, E. (2013). *Diseño e integración en Android de un sistema de realidad aumentada y reconocimiento de imágenes para un sistema de domótica asistencial*. http://e-archivo.uc3m.es/handle/10016/17659

Cómo guardar contenido en una base de datos local con Room. (n.d.). Retrieved November 23, 2021, from https://developer.android.com/training/data-storage/room

Dong, E., Du, H., & Gardner, L. (2020). An interactive web-based dashboard to track COVID-19 in real time. In *The Lancet Infectious Diseases* (Vol. 20, Issue 5, pp. 533–534). Lancet Publishing Group. https://doi.org/10.1016/S1473-3099(20)30120-1

García, A. (2012). *INTELIGENCIA ARTIFICIAL. Fundamentos, práctica y aplicaciones*. RC Libros. https://books.google.com.co/books?id=WDuqquRP70UC

Home - OpenCV. (n.d.). Retrieved November 14, 2021, from https://opencv.org/

Ismael Pineda Palencia. (2019). *Aplicación de recomendaciones de moda basada en redes de aprendizaje profundo*. https://riuma.uma.es/xmlui/bitstream/handle/10630/18993/PinedapalenciaismaelMemoria.pdf?sequence=1&isAllowed=y

Jose Luis Villaluenga Morán. (2019). *OpenStreetCam: reconocimiento automático de objetos en imágenes mediante machine learning*. http://openaccess.uoc.edu/webapps/o2/bitstream/10609/88287/6/jvillaluengaTFG0119memoria.pdf

Jurado García, M. E., & Padilla Porras, A. F. (2018). *Sistema de reconocimiento facial con redes neuronales para la toma de asistencia en aulas de clase*. 22.

JVM Explained | Java Tutorial Network. (2021). https://javatutorial.net/jvm-explained

Martínez Gandia, J. A. (2020). *Propuesta de diseño de prototipo para el control de aforo y el distanciamiento social en Institución Educativa de Educación Superior Tecnológica en la ciudad de Barranquilla*. https://bonga.unisimon.edu.co/handle/20.500.12442/6905

Méndez, D. (2015). *Introducción a Android Studio | Desarrolladores de Android*. Developers. https://developer.android.com/studio/intro?hl=es-419

Montiel, H. (2015). Uso de redes neuronales para el reconocimiento de rostros en ambientes controlados. *Tecnura*, *19*(0), 67–77. https://doi.org/10.14483/22487638.10373

OpenCV: Clasificador en cascada. (n.d.). Retrieved November 14, 2021, from https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

Oracle. (2010). *Lesson: Object-Oriented Programming Concepts (The Java™ Tutorials > Learning the Java Language)*. http://docs.oracle.com/javase/tutorial/java/concepts/

Rogé, J., El Zufari, V., Vienne, F., Ndiaye, D., Zhang, H., Lu, Y., Gupta, S., Zhao, L.,

Bustamante, J. C., Salgado, S. S., Holyoak, K. J., Thagard, P., Es, L. O. S. E. D. E. E., García García, P. P., Gonzalez, J. C., Varela, J. A., De, T., & Electrónico, D. I. (2013). Reconocimiento de imágenes utilizando redes neuronales artificiales. *Estudios Gerenciales*, *13*(3), 1017–1030. http://doi.wiley.com/10.1016/0364-0213(89)90016-5%5Cnhttp://linkinghub.elsevier.com/retrieve/pii/S0123592315000327%5Cnhttp://dx.doi.org/10.1016/j.im.2014.07.005%5Cnhttp://linkinghub.elsevier.com/retrieve/pii/S092575351500123X

Rouhiainen, L. (2018). *INTELIGENCIA ARTIFICIAL 101 COSAS QUE DEBES SABER HOY SOBRE NUESTRO FUTURO INTELIGENCIA ARTIFICIAL*. www.planetadelibros.com

sqlite. (2018). *Página de inicio de Matt*. http://chrome.ws.dei.polimi.it/index.php?title=Matt%27s_Home_Page

Teigens, V. (2020). *Inteligencia Artificial General*. Cambridge Stanford Books. https://books.google.com.pe/books?id=4R3NDwAAQBAJ

towardsdatascience. (2018). *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science*. Towards Data Science. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Web, A., Software, U., Carlos, L., & Andrade, G. N. (2012). Autenticación Por Reconocimiento Facial Para. *Instname:Universidad Pontificia Bolivariana*. https://repository.upb.edu.co/handle/20.500.11912/2063%0Ahttps://repository.upb.edu.co/bitstream/handle/20.500.11912/2063/digital_24318.pdf?sequence=1&isAllowed=y

ANEXOS

Anexo A Preguntas y resultados encuesta

Figura 36 Encuesta pregunta 1

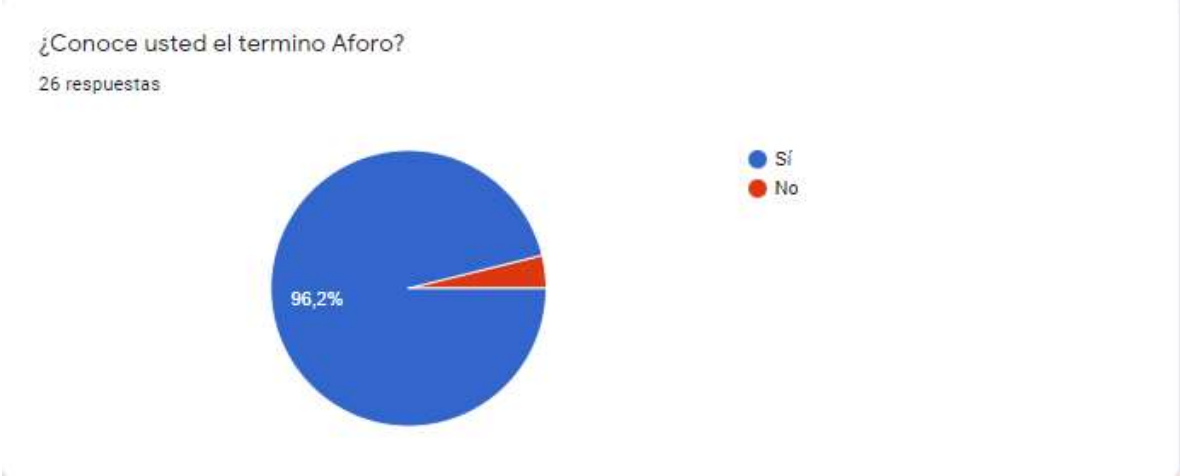


Figura 37 Encuesta pregunta 2

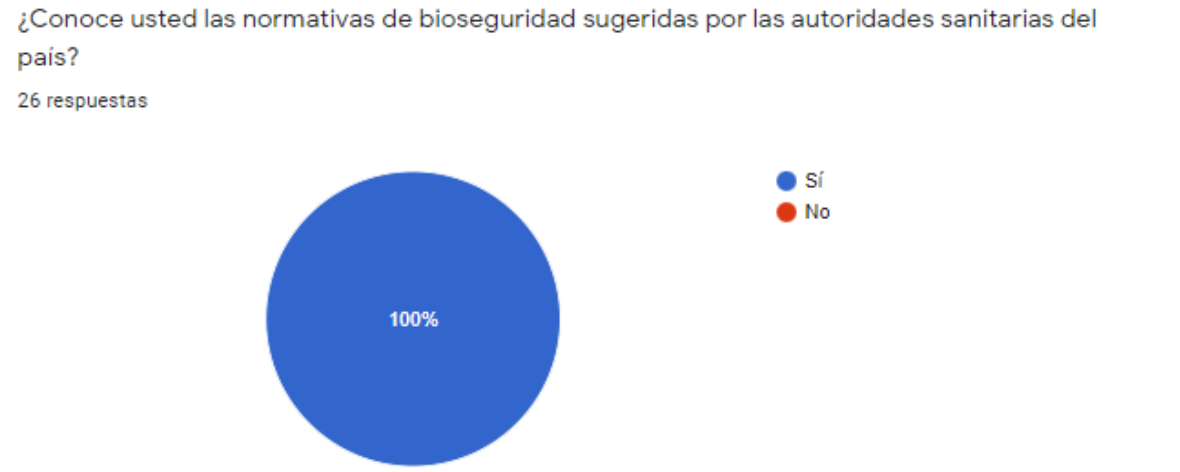


Figura 38 Encuesta pregunta 3

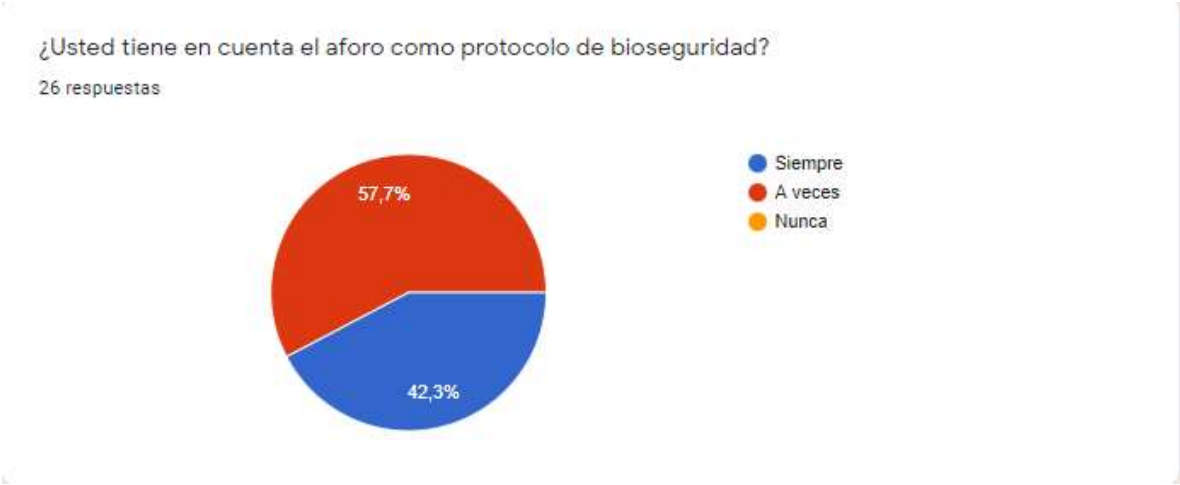


Figura 39 Encuesta pregunta 4

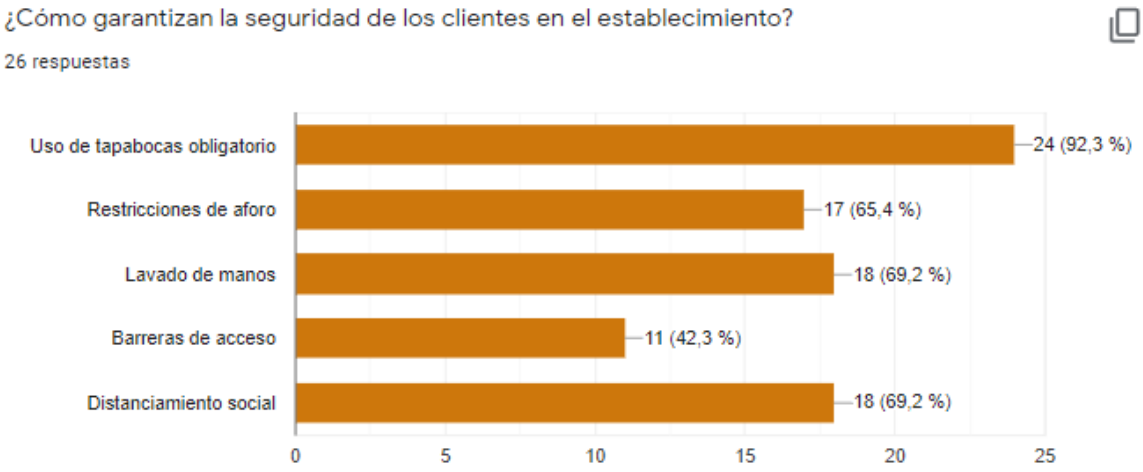


Figura 40 Encuesta pregunta 5

Figura 41 Encuesta pregunta 6

Figura 42 Encuesta pregunta 7

Figura 43Encuesta pregunta 8

Figura 44 Encuesta pregunta 9

Figura 45 Encuesta pregunta 10

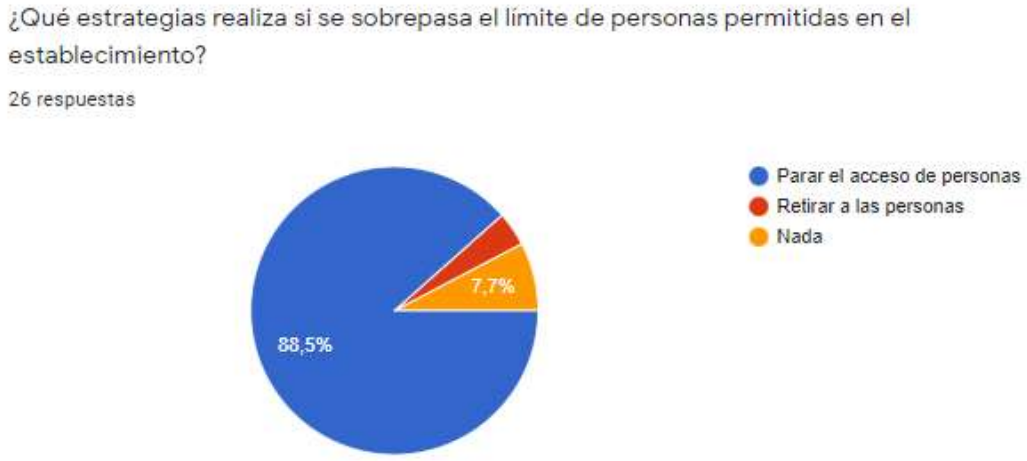


Figura 46 Encuesta pregunta 11

Anexo B Documento de arquitectura

En este documento se encuentran la especificación de los mockups y los diagramas de diseño del proyecto.

ARQUITECTURA DEL SOFTWARE

Software: AJS203

Ficha del Documento

Fecha	Autor	Verificado
20/08/2021	Juan Sebastián Velásquez	

Documento Validado por las partes en agosto del 2021

Director

Desarrollador

Juan Sebastián Velásquez Manzano

Introducción

Se presenta el documento de arquitectura del software que se utilizará para la creación de la solución informática denominado **Aplicación móvil para identificación y conteo de personas**, que será una plataforma de software diseñado para realizar un control de aforo soportado en dispositivos móviles utilizando reconocimiento de imágenes e inteligencia artificial, con el fin de identificar y contar personas en un establecimiento comercial automáticamente.

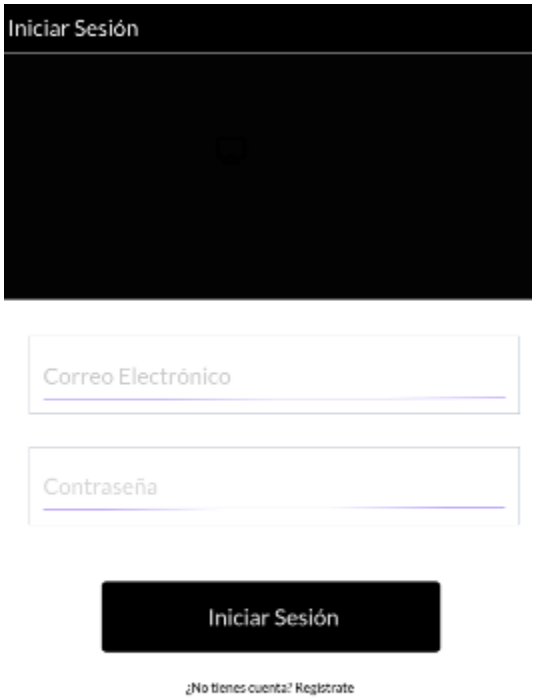
Arquitectura

Interfaces(Mockups)

Inicio de sesión

En esta interfaz se le muestra al usuario dos campos de texto, en el primero debe ingresar su correo electrónico y en el segundo debe ingresar la contraseña. Una vez ingresados esos datos y al darle al botón “Iniciar Sesión” se redirige la página principal de la aplicación.

Figura 47 Interfaz inicio de sesión



Fuente: Propia

Registro

En esta interfaz se habilitan los campos para pedirle la información necesaria a usuario y que quede registrado en el sistema. Entre las cuales se encuentran: “Nombre de usuario”, “Contraseña”, “Dirección”, entre otras.

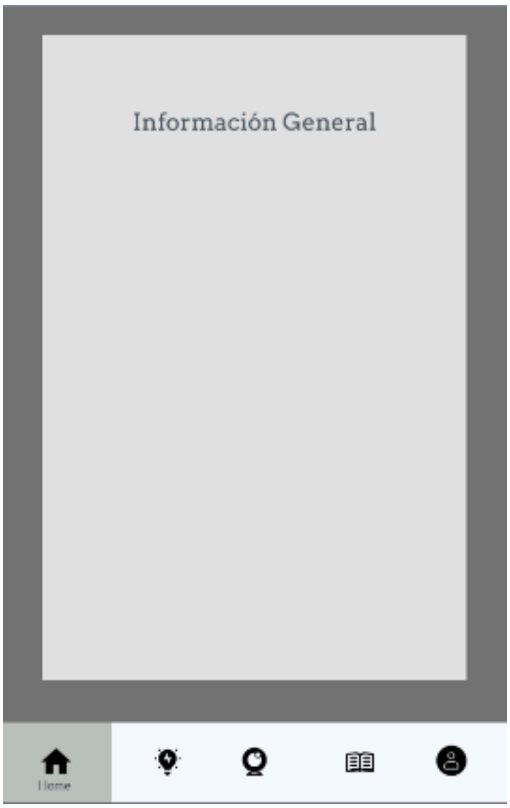
Figura 48 Interfaz de registro

Fuente: Propia

Home

En esta interfaz se le mostrará al usuario una breve descripción del funcionamiento de la app y su alcance. Además, en la parte inferior hay unos botones que le permite moverse por la diferentes funcionalidades del sistema, entre ellas tenemos los botones: “Home”, “Instrucciones”, “Cámara”, “Registros” y “Perfil”, cada una con sus respectivos logos.

Figura 49 Interfaz Home



Fuente: Propia

Cámara

En esta interfaz se le permite al usuario tomar la respectiva foto en el lugar donde desee llevar el control de aforo, todo por medio de un botón en el centro. Además, se le da la opción de diligenciar el aforo que se esté manejando en ese momento.

Figura 50 Interfaz cámara

Fuente: Propia

Validación foto tomada

En esta interfaz se le muestra al usuario la foto ya tomada, en ella se pueden visualizar las personas identificadas por medio de un recuadro en el rostro. Además, hay unas cajas de texto que indican el aforo y el número de personas que contó el sistema en la foto.

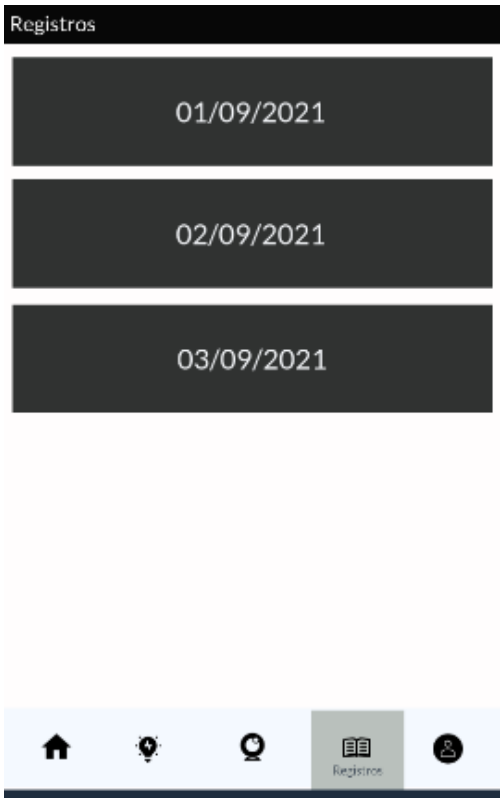
Figura 51 Interfaz de detalle

Fuente: Propia

Registros

En esta interfaz se muestra al usuario una lista de los días en los que se realizó el control de aforo. Cada uno de estos los redirige hasta la especificación de dichos registros.

Figura 52 Interfaz de registros por día



Fuente: Propia

Detalle

En esta interfaz el usuario puede visualizar los registros a detalle que ocurrieron en un determinado día, cada item posee: Nombre del local, fecha, hora, Cantidad máxima de personas y la foto tomada. Además, contiene un botón que permite filtrar los datos que se hayan sobrepasado el aforo.

Figura 53 Interfaz de listado de detalles

Fuente: Propia

Prototipo interactivo(mockup):

https://marvelapp.com/prototype/h1hgaih/screen/81799440 .

Modelo relacional

En este modelo se plantearon cada una de las entidades que componen el sistema junto con sus respectivas relaciones.

- Usuario: se guardan los atributos correspondientes al usuario: Nombre de usuario, foto del local, dirección, capacidad de aforo y contraseña.
- Registro: se guardan cada una de las fechas en que se realizó un control de aforo.
- Detalle: se guardan los datos sacados de la toma de la imagen: La hora en que se tomó la foto, La cantidad de personas identificadas, la capacidad que se está manejando y la respectiva imagen tomada.

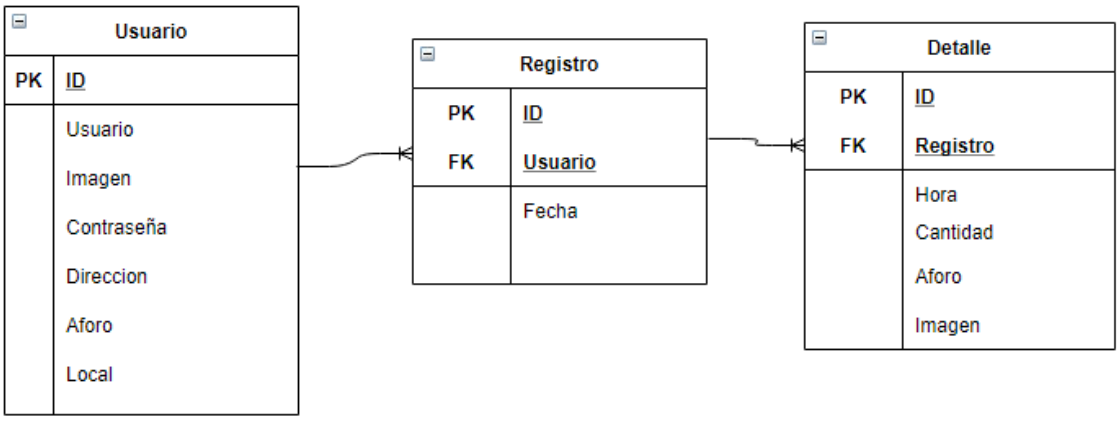
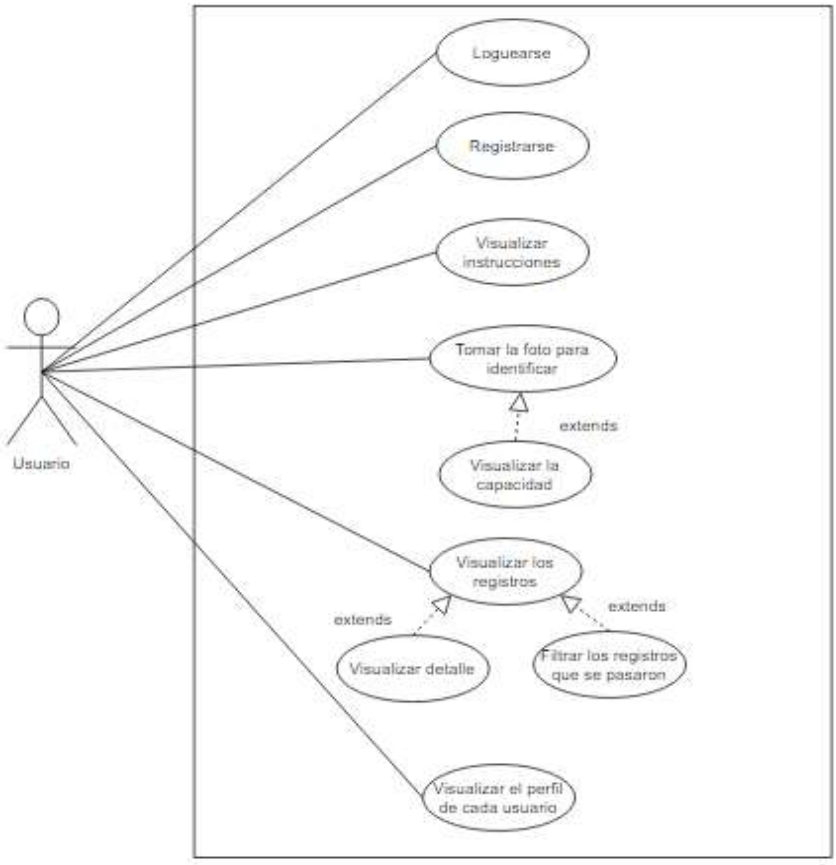


Diagrama de casos de uso

En este diagrama se plantearon todos los actores con sus respectivos casos de usos los cuales afectan o intervienen en el sistema. Se puede ver en el recuadro los casos de uso que contiene el usuario con respecto a las funcionalidades del sistema.



Especificación:

Tabla 11 Caso de uso 1

Caso de uso:	Loguearse
Actor:	Usuario
Precondición:	Haber ingresado a la aplicación
Postcondición:	Entrar al inicio de la aplicación
Flujo de eventos básico	
1.	El usuario ingresa la información pertinente de ingreso
2.	El sistema valida si el usuario está registrado

3.	
Flujo de eventos Alternativo	
5.	El sistema muestra una alerta que indica que los datos son erróneos

Tabla 12 Caso de uso 2

Caso de uso:	Registrarse
Actor:	Usuario
Precondición:	Haberse registrado con anterioridad
Postcondición:	Ir a la pantalla de login
Flujo de eventos básico	
1.	El sistema muestra la información
2.	El usuario guarda sus datos
3.	El sistema guarda la información en el base de datos

Tabla 13 Caso de uso 3

Caso de uso:	Visualizar instrucciones
Actor:	Usuario
Precondición:	Haberse registrado con anterioridad
Postcondición:	Mostrar la información solicitada
Flujo de eventos básico	
1.	El sistema muestra la información de las instrucciones de uso
2.	El visualiza las instrucciones de la app

Tabla 14 Caso de uso 4

Caso de uso:	Tomar fotos para identificar.
Actor:	Usuario.
Precondición:	Haber aceptado permisos del uso de cámara.

Postcondición:	Visualizar el aforo según la foto tomada.
Flujo de eventos básico	
1.	El sistema solicita el número máximo de personas.
2.	El usuario escribe el número máximo de personas.
3.	El usuario toma la foto.
4.	El sistema procesa la información.
5.	El sistema guarda los datos

Tabla 15 Caso de uso 5

Caso de uso:	Visualizar la capacidad
Actor:	Usuario
Precondición:	Tomar fotos para identificar.
Postcondición:	Aforo validado
Flujo de eventos básico	
1.	El sistema muestra los datos procesados.
2.	El sistema identifica y cuenta las personas que aparecen en la foto
3.	El usuario puede visualizar a las personas señaladas y el aforo
Flujo de eventos Alternativo	
1.	El sistema muestra una alerta si se pasó el número máximo de personas.

Tabla 16 Caso de uso 6

Caso de uso:	Visualizar registros
Actor:	Usuario
Precondición:	Tener registros de aforo almacenados en el teléfono
Postcondición:	Visualizar los registros a detalle
Flujo de eventos básico	

1.	El sistema muestra la información agrupada por medio días.
2.	El usuario puede visualizar dicha información.
3.	El sistema da la opción de mostrar la información adquirida durante ese día.
4.	El usuario puede ingresar en alguno de esos registros.

Tabla 17 Caso de uso 7

Caso de uso:	Visualizar los registros a detalle
Actor:	Usuario
Precondición:	Seleccionar el detalle de un registro del día
Postcondición:	Mostrar la información y tener la opción de filtro disponible
Flujo de eventos básico	
1.	El sistema muestra los análisis de aforo realizados durante del día
2.	El usuario puede visualizar la información de los momentos del día
3.	El usuario puede desplazarse a través de los diferentes registros realizados durante el día

Tabla 18 Caso de uso 8

Caso de uso:	Filtrar los registros que se pasaron
Actor:	Usuario
Precondición:	Visualizar el detalle de los registros del día
Postcondición:	Mostrar la información solicitada
Flujo de eventos básico	
1.	El sistema da la opción de filtrar los registros que se pasaron.
2.	El usuario puede filtrar los registros
3.	El usuario puede desplazarse a través de los diferentes registros realizados que se pasaron durante el día.

Tabla 19 Caso de uso 9

Caso de uso:	Filtrar los registros que se pasaron
Actor:	Usuario
Precondición:	Visualizar el detalle de los registros del día
Postcondición:	Mostrar la información solicitada
Flujo de eventos básico	
1.	El sistema da la opción de filtrar los registros que se pasaron.
2.	El usuario puede filtrar los registros
3.	El usuario puede desplazarse a través de los diferentes registros realizados que se pasaron durante el día.

Tabla 20 Caso de uso 10

Caso de uso:	Visualizar perfil
Actor:	Usuario
Precondición:	Haberse registrado con anterioridad
Postcondición:	Loguearse
Flujo de eventos básico	
1.	El sistema muestra la información del perfil del usuario
2.	El usuario visualiza su perfil
3.	El sistema da la opción de cerrar sesión
Flujo alternativo	
4.	El usuario tiene la opción de cerrar sesión

Diagrama de clases

Este diagrama nos muestra cada una de las clases que componen nuestro sistema y sus relaciones, entre ellas tenemos:

- Usuario: Son las personas que van a ingresar y utilizar el sistema
- Registro: Son los registros que se realizaron durante el día
- Detalle: Son los detalles asociados a cada uno de los registros ocurridos durante el día, además filtra los datos con base al aforo.

Figura 54 Diagrama de clases

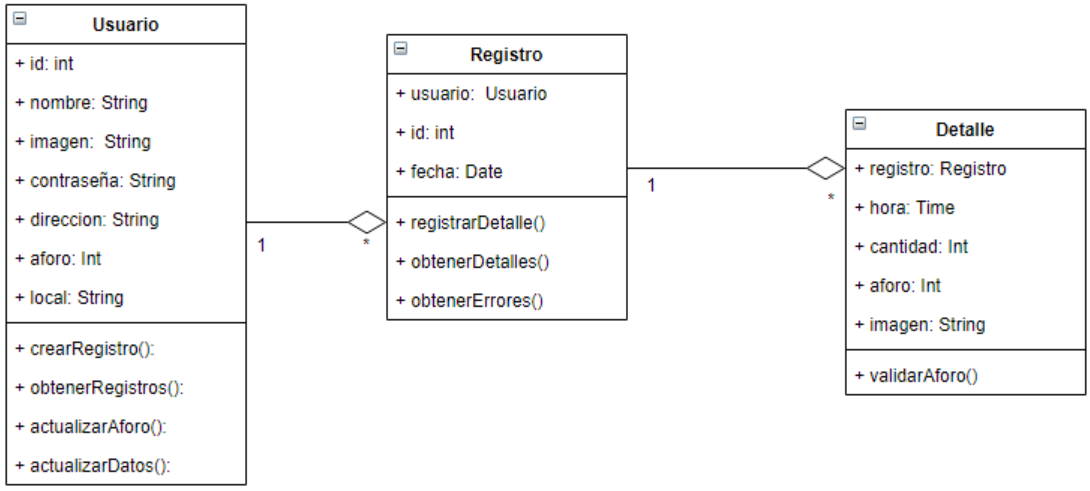


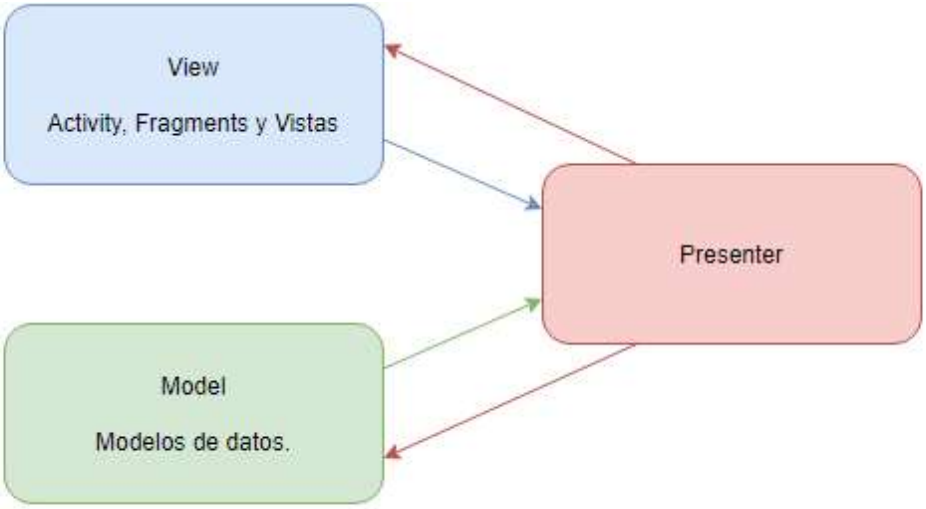
Diagrama de arquitectura MVP

La arquitectura planteada para este proyecto es el patrón de diseño MVP (Modelo Vista Presentador).

- Vista: Dentro del entorno se elaborará una interfaz visual interactiva (Vista), que contará con diferentes clases y vistas como, por ejemplo:
    - Vistas: Login, Registro, Inicio, Instrucciones, cámara, listado de registros, detalle y perfil.
    - Activitys y frament relacionados a estas vistas

- Presenter: Es la parte que recibe los datos y los valida, originados por la vista y así como el encargado del flujo de datos por medio de clases que implementan la vista y el modelo.
- Modelo: Es la información que se muestra y genera (Lógica del negocio), esta se obtiene de la comunicación con la base de datos (ROOM), esta contiene:
    - Entidades: usuario, registro y detalle.
    - Dao: UsuarioDao, RegistroDao y DetalleDao.
    - La base de datos.

Figura 55 Diagrama MVP



Anexo C Escenario de pruebas

En este documento se describirán en detalle cada uno de los escenarios de pruebas que se hayan identificado como necesarios para verificar la funcionalidad completa del sistema.

PLAN DE PRUEBAS

Software: AJS203

Ficha del Documento

Fecha	Autor	Verificado
20/08/2021	Juan Sebastián Velásquez	

Documento Validado por las partes en Noviembre del 2021

Director Desarrollador

Juan Sebastián Velásquez Manzano

Introducción

1. Objetivo

Este documento tiene como finalidad recoger los datos de las pruebas sacados de cada uno de los escenarios para verificar que el sistema satisface los requisitos especificado para la solución informática denominada **Aplicación móvil para identificación y conteo de personas**.

Escenarios de prueba

En este apartado se describirán en detalle cada uno de los escenarios de pruebas que se hayan identificado como necesarios para verificar la funcionalidad completa del sistema.

Prueba 1	Registro del usuario
Descripción	Registro del usuario al sistema
Estado inicial	La aplicación debe estar instalada
Estado final	El sistema guarda la información del usuario y la imagen ingresada
Procedimiento	Se van a crear 5 usuarios con diferentes datos para verificar que se agreguen correctamente cada uno de los registros.
Resultado Obtenido	Éxito

Tabla 21 Datos de prueba usuario

Datos						
Nombre Usuario	**Nombre Local**	**Dirección**	**Foto del local**	**Aforo**	**Contraseña**	**Resultado**
Sebastian12	Restaurante Otero	Calle 11# 35-10	Sí	25	123456789	Éxito
JoseAlejo23		Call 20 #2-13	Sí	10	‘sdsd_	Éxito
AndreaR	Restaurante Ginza	Car31 # 1-09	No	15	35498454	Éxito
JuanR	Dulce pecado heladería	Car2 # 15-30	Sí	15	1	Éxito
AndresG	Market Acacias	Car20 # 12-01	No	5	35498454	Éxito

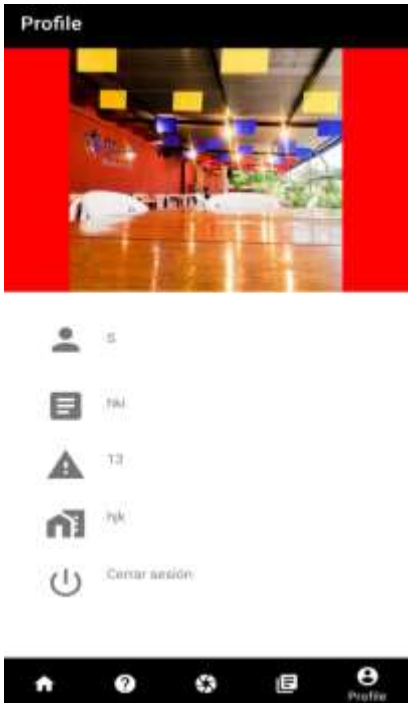
Figura 56 Prueba de registro Usuario #1

Prueba 2	Login
Descripción	Login para que el usuario ingrese al sistema
Estado inicial	Haberse registrado con anterioridad
Estado final	Sesión iniciada
Procedimiento	Se va a intentar loguearse los 5 usuarios anteriormente creados
Resultado Obtenido	Éxito

Figura 57 Prueba de login Usuario #1

Prueba 3	Perfil
Descripción	Login para que el usuario ingrese al sistema
Estado inicial	Debe estar logueado en sistema
Estado final	Se visualizan todos los datos del usuario en el perfil.
Procedimiento	Se verifica que en cada uno de los usuarios creados se muestren todos sus datos y que la imagen cargue correctamente. Además de que la opción de cerrar sesión funcione bien.
Resultado Obtenido	Éxito

Figura 58 Prueba de perfil usuario #1



Prueba 4	Identificación y conteo de personas manual
Descripción	Realiza la toma de las fotos, identificación de las personas y el conteo de las personas identificadas manualmente.
Estado inicial	• Cámara activada • Aforo diligenciado • Estado “ninguno”
Estado final	• Toma de la foto • Guardar la imagen en el dispositivo • Identifica las personas en la imagen • Se visualiza todos los datos en el detalle
Procedimiento	Se van realizan 5 tomas manuales en diferentes lugares, perfiles, y con aforos diferentes para comprobar que se realice la identificación y conteo de manera eficiente.
Resultado Obtenido	• Todas las imágenes se guardan correctamente con sus respectivos registros en la base de datos. • Se listan los datos referentes a la extracción de la imagen

	• Identifica a las personas en el mayor de los casos i hay buena iluminación • Identifica las personas con tapabocas, pero con un porcentaje ya mucho menor.

Tabla 22 Datos de tomas manuales

# De toma	Descripción	Aforo	identificación	Registro y detalle	¿Imagen Cargada?	Conteo
1	Frontal	5	1/1	Correcto	Sí	1
2	Perfil	2	1/1	Correcto	Sí	0
3	+ de una persona	2	2/2	Correcto	Sí	2
4	Tapabocas	5	1/1	Correcto	Sí	1
5	Contraluz	5	0/2	Correcto	Sí	0

- Evidencia de los tomas realizadas:

Figura 60 Toma manual: frontal

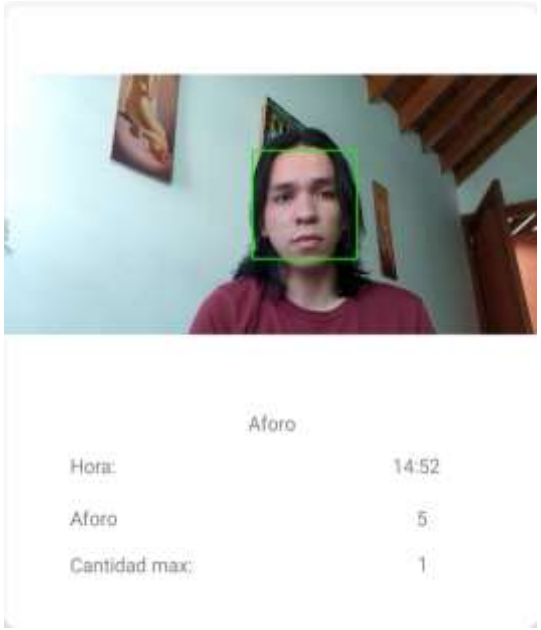


Figura 59 Toma manual: tapacabocas

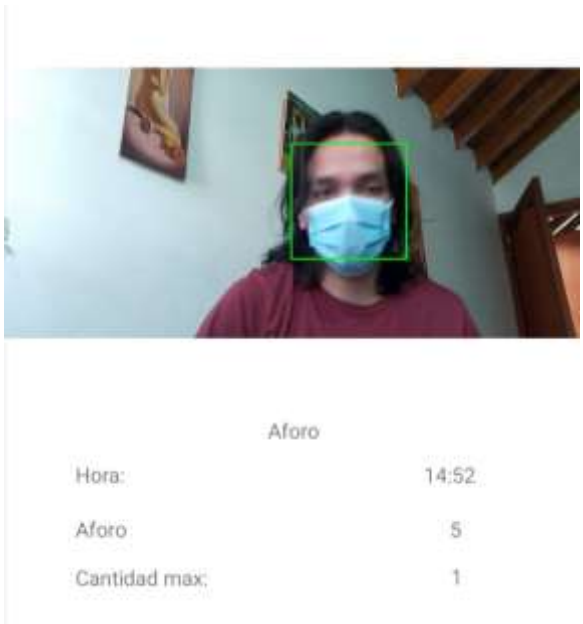


Figura 62 Toma manual: Perfil

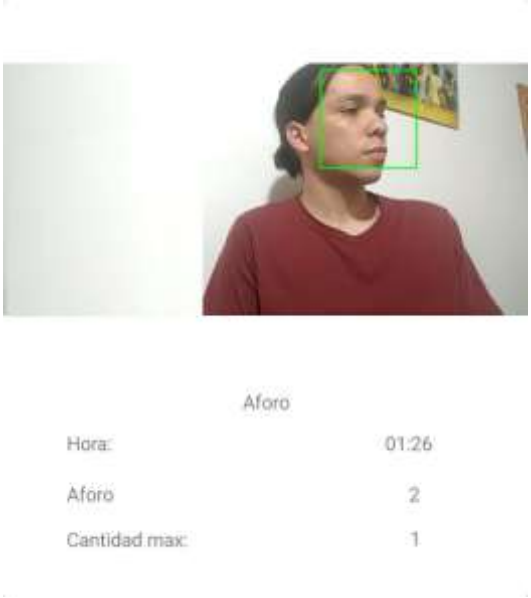
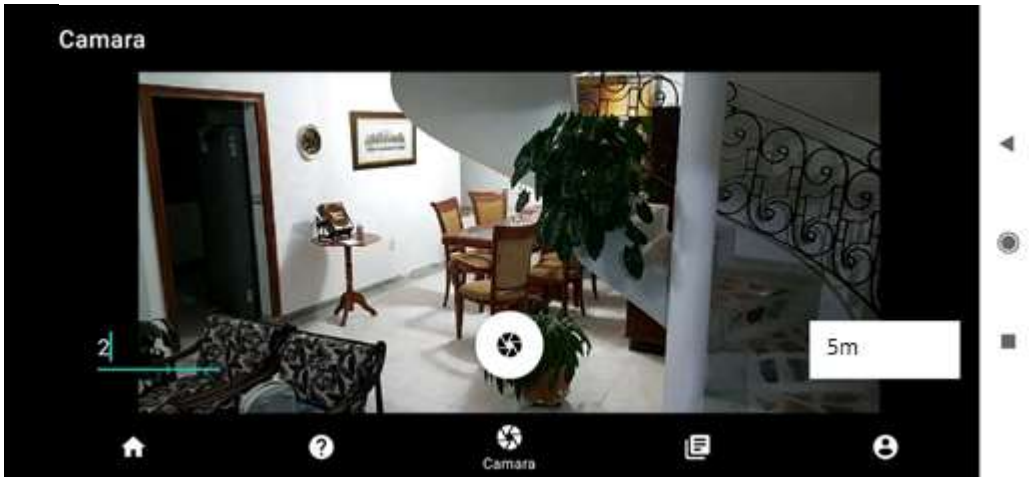


Figura 61 Toma manual: Contraluz

Figura 63 Toma manual: Mas de una persona

Prueba 5	Identificación y conteo de personas automáticamente
Descripción	Realiza la toma de las fotos, identificación de las personas y el conteo de las personas de manera automática, tomando en cuenta el tiempo establecido por el usuario (5s, 1m, 5m o 10m).
Estado inicial	• Cámara activada. • Aforo diligenciado. • Tiempo seleccionado.
Estado final	• Toma de la foto cada cierto tiempo(tempo). • Guardar las imágenes en el dispositivo y las lista en el detalle. • Identifica las personas en la imagen.
Procedimiento	Se va a realizar la identificación y el conteo de manera automática en un entorno seleccionado durante tres horas, teniendo en cuenta que el tiempo base es de 5m.
Resultado Obtenido	Dado que el dispositivo se ubicó en un lugar amplio y con buena iluminación en la mayoría de los registros en los que hay una persona de por medio las logra identificar, teniendo en cuenta los datos un porcentaje identificación de 80%.

Figura 64 Lugar de prueba seleccionado para la identifiación manual



Registros de personas identificadas durante las 3 horas:

Figura 66 Toma automática: Registro 1

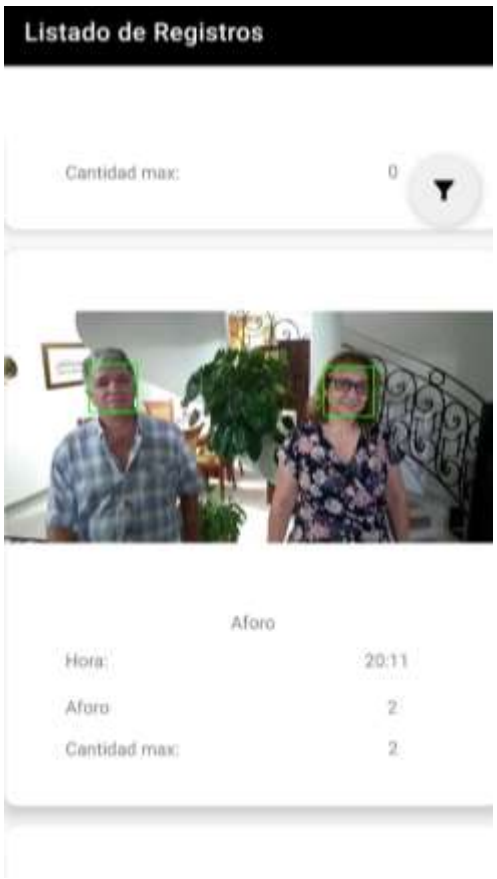


Figura 65 Toma automática: Registro 2

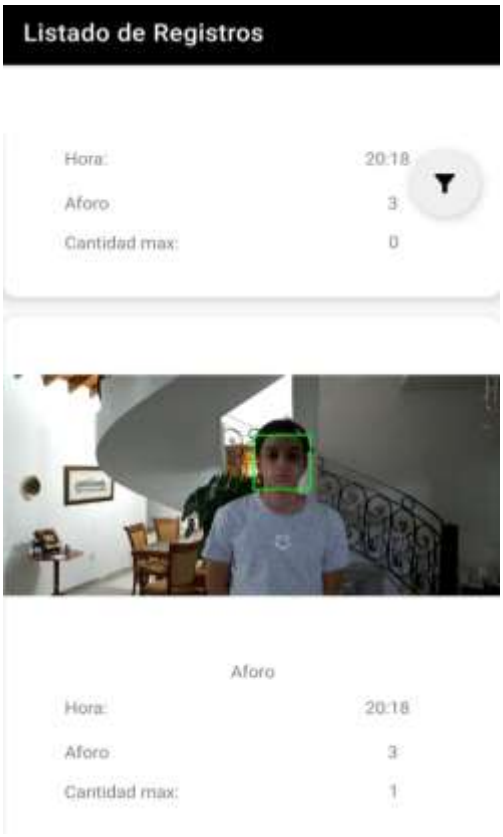


Tabla 23 Prueba funcionalidad registros por día

Prueba 6	Guardar registros del día
Descripción	Lista de registros diarios que contienen todos los detalles (registros de tomas) realizados durante el día
Estado inicial	Haber realizado alguna toma de control de aforo, ya sea manual o automática.
Estado final	Carguen los datos de cada item y se listen cada uno.
Procedimiento	Tomando en cuenta los registros realizados en pruebas anteriores, se verifican que se cree el registro diario correspondiente.
Resultado Obtenido	Éxito.

Fuente: Propia

Tabla 24 Prueba funcionalidad detalle

Prueba 7	Guardar detalles de los datos adquiridos.
Descripción	registros y visualización del detalle de los datos extraídos al momento de tomar la foto
Estado inicial	Haber realizado alguna toma de control de aforo, ya sea manual o automática.
Estado final	Se guardan los datos y se visualizan en el detalle de cada registro.
Procedimiento	Tomando en cuenta los registros realizados en pruebas anteriores, se verifican que se guarden cada uno de los datos en los detalles y se visualicen en un listado.
Resultado Obtenido	Éxito.

Fuente: Propia