ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Plataforma de monitoreo de vehículos en parqueaderos de ESPOL

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniero en Telemática

Presentado por:

Andrés Antonio Contreras Calle

Alfonso David Enríquez Salguero

GUAYAQUIL - ECUADOR

Año: 2024

DEDICATORIA

Andrés Antonio Contreras Calle

Dedico este proyecto en primer lugar a Dios por darme la perseverancia y sabiduría para continuar con mis estudios y lograr el objetivo trazado. A mi familia, en especial a mi madre, que siempre estuvo a mi lado y por ser el pilar fundamental, motivándome siempre a seguir adelante con su ejemplo de responsabilidad y sacrificio.

Alfonso David Enríquez Salguero

Dedico este proyecto a Dios por guiar mis pasos y darme fuerzas para continuar con mis objetivos. A mi familia, especialmente a mi madre, quien por ella soy lo que soy como persona, por su apoyo incondicional en todas las etapas de mi vida, quien ha velado por mi bienestar y educación, y por todas sus enseñanzas y consejos para lograr mis objetivos.

AGRADECIMIENTOS

Andrés Antonio Contreras Calle

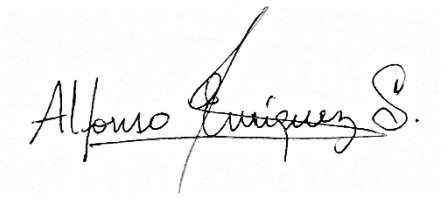
Agradezco a mis profesores y compañeros que colaboraron con sus conocimientos, brindando sus aportes y experiencias para lograr el objetivo planteado.

Alfonso David Enríquez Salguero

Mi sincero agradecimiento a la ESPOL, por contar con una educación de calidad, a mis profesores por compartir sus conocimientos y experiencias en el transcurso de mi vida estudiantil, a mis amigos y a todas las personas que me acompañaron y motivaron en la realización de este proyecto.

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; Andrés Antonio Contreras Calle y Alfonso David Enríquez Salguero damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual”



Andrés Antonio Contreras Calle

Alfonso David Enríquez Salguero

EVALUADORES

Firmado electrónicamente por:
IGNACIO MARIN GARCIA

Ignacio Marín Garcia

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
RAYNER STALYN DURANGO ESPINOZA

Rayner Durango Espinoza

PROFESOR TUTOR

RESUMEN

El monitoreo de aforo de vehículos es un tema de interés que ha surgido dentro de la Escuela Superior Politécnica del Litoral (ESPOL). La población estudiantil de este centro educativo se mantiene en constante crecimiento, lo cual ha generado que la cantidad de vehículos que ingresan se incremente, lo que a su vez genera ciertos problemas en diversos sectores de la institución como aglomeraciones en parqueaderos. Este estudio analiza la situación actual en la ESPOL, identificando parámetros de medición y evaluación para buscar una solución a la problemática que se presenta. Mediante la implementación de un sistema de identificación de aforo se puede determinar la cantidad vehículos que se encuentra en un área específica. Se recopila la información histórica del monitoreo de aforo en un servidor de base de datos para que pueda ser utilizada en análisis estadísticos posteriormente.

A través del uso de cámaras y un algoritmo especializado se monitorea el ingreso y salida del flujo de vehículos en los parqueaderos. Esto se logró mediante un script desarrollado con el lenguaje de programación Python, el cual realiza el conteo por identificación de movimiento. Esta información es enviada a una plataforma web vinculada a una aplicación móvil donde las personas tienen acceso para ver el aforo actual en los puntos de interés. De esta manera, los usuarios pueden aparcar su vehículo en el sector que mejor se adapte a sus necesidades.

Con esta plataforma online de monitoreo, los estudiantes, docentes y personal administrativo de la ESPOL visualizan en “tiempo real” el aforo de los parqueaderos para optimizar su tiempo a la hora de aparcar y así reducir el consumo de combustible.

Palabras Clave: monitoreo, aglomeración, algoritmo, plataforma online.

ABSTRACT

The monitoring of the capacity of people and vehicles is a topic of interest that has arisen within Escuela Superior Politécnica del Litoral (ESPOL). The student population of this educational center is constantly growing, which generates certain problems in various sectors of the institution such as crowds in parking lots, dining rooms and study sites such as libraries. This study analyzes the current situation in ESPOL, identifying measurement and evaluation parameters to find a solution to the problem that arises. By implementing a capacity identification system, it is possible to determine the number of vehicles that are in a specific area. Historical capacity monitoring information is collected in a database server so that it can be used in later statistical analysis.

Using cameras and a specialized algorithm, the entry and exit of vehicles in parking lots are monitored. This was achieved through a script developed with the Python programming language, which performs the count by movement identification. This information is sent to a web platform to which people have access to see the current capacity at the points of interest. In this way, users can determine the sector that best suits their needs.

With this online monitoring platform, ESPOL students, teachers and administrative staff view the parking capacity in “real time” to optimize their time when parking and thus reduce fuel consumption.

Keywords: monitoring, agglomeration, algorithm, online platform.

INDICE GENERAL

ABREVIATURAS

ABE	Encriptación Basada en Atributos (del inglés Attribute Based Encryption)
API	Interfaz de Programación de Aplicaciones (del inglés Application Programming Interfaces)
AVL	Localización Automática de Vehículos (del inglés Advanced Vehicle Location)
ESPOL	Escuela Superior Politécnica del Litoral
FADCOM	Facultad de Arte, Diseño y Comunicación Visual
FCM	Mitigación de Colisión Frontal (del inglés forward collision mitigation)
FCNM	Facultad de Ciencias Naturales y Matemáticas
FCSH	Facultad de Ciencias Sociales y Humanísticas
FCV	Facultad de Ciencias de la Vida
FICT	Facultad de Ingeniería en Ciencias de la Tierra
FIEC	Facultad de Ingeniería en Electricidad y Computación
FIMCM	Facultad de Ingeniería en Marítima y Ciencias del Mar
FIMCP	Facultad de Ingeniería en Mecánica y Ciencias de la Producción
GIS	Sistema de Información Geográfica (del inglés Geographical Information System)
GPRS	Servicio General de Paquetes de Radio (del inglés General Packet Radio Service)
GPS	Sistema de Posicionamiento Global (del inglés Global Positioning System)
GSM	Sistema Global para las Comunicaciones Móviles (del inglés Global System For Mobile Comunication)
HTTP	Protocolo de Transferencia de Hipertexto en españo (del inglés Hypertext Transfer Protocol)
IDE	Entorno de Desarrollo Integrado (del inglés Integrated Development Environment)
IoT	Internet de las cosas (del inglés Internet Of Things)
IP	Protocolo de Internet (del inglés Internet Protocol)
MAC	Control de Acceso de Medios (del inglés Media Access Control)
MATLAB	Laboratorio de Matrices (del inglés MATrix LABoratory)

MQTT	Protocolo de mensajería ligero (del inglés Message Queing Telemetry Transport)
PSO	Optimización por Enjambres de Partículas (del inglés Particle Swarm Optimization)
REST	La transferencia de estado representacional (del inglés Representational State Transfer)
RFID	Identificación por radiofrecuencia (del inglés Radio Frequency Identification)
SIFT	Transformación de característica invariante de escala (del inglés Scale Invariant Feature Transform)
SURF	Funciones robustas aceleradas (del inglés Speeded Up Robust Features)
TLS	La seguridad de la capa de transporte (del inglés Transport Layer Security)
TPDA	Tráfico Promedio Diario Anual
USB	Bus Universal en Serie (del inglés Universal Serial Bus)

ÍNDICE DE FIGURAS

CAPITULO 1

1. INTRODUCCION

En el Ecuador existen diversas universidades y escuelas politécnicas registradas, siendo la Escuela Superior Politécnica del Litoral una de las mejores universidades públicas del país gracias a su amplia trayectoria en desarrollo y difusión de la innovación e investigación científica (ESPOL, 2023). Es por ello por lo que cada año los jóvenes optan por estudiar su carrera profesional en esta prestigiosa institución de educación superior lo que conlleva a que su población estudiantil se encuentre en constante crecimiento ocasionando ciertos inconvenientes en diferentes sectores del campus universitario (El Universo, 2023).

El proyecto consiste en obtener datos válidos que ayuden a determinar la cantidad de vehículos y el nivel de afluencia que tiene un sitio determinado o punto de interés utilizando recursos tecnológicos que permitan la recolección de dichos datos de manera eficiente y confiable. Gracias al avance de la tecnología, los datos pueden ser recopilados con dispositivos ubicados de forma estratégica y ser enviados a través de Internet para luego ser desplegados en una aplicación móvil disponible para teléfonos inteligentes, y de esta manera visualizar el estado y funcionamiento del sistema en “tiempo real”.

Así, el proyecto integrador comienza explicando la definición del problema a resolver los objetivos a alcanzar, la información conceptual necesaria para comprender la terminología utilizada a lo largo del estudio, un análisis descriptivo de proyectos similares, la metodología empleada en el desarrollo del proyecto, la evaluación de los resultados y finalmente un análisis detallado que direccionen a la obtención de conclusiones y toma de decisiones adecuadas en base a los datos recopilados.

1.1. Descripción del problema

La ESPOL es una institución pública de educación superior que nació para satisfacer las crecientes demandas de instrucción científica-técnica en el país acogiendo a cientos de aspirantes cada año y convirtiéndose en una de las primeras opciones en los jóvenes ecuatorianos al momento de postular por una carrera universitaria. Esto genera una gran afluencia de personas que incluye al cuerpo estudiantil, docente y administrativo, lo cual causa efectos adversos en diferentes áreas del establecimiento universitario. La mayor parte de los estudiantes politécnicos asisten a sus clases durante la jornada matutina, lo cual produce congestionamiento vehicular en los parqueaderos del campus.

1.2. Justificación del problema

Con el crecimiento poblacional estudiantil, sumado al incremento de vehículos en la ciudad, da origen a problemas dentro del campus al momento de buscar parqueo no solo para estudiantes sino también para el personal administrativo. En ocasiones las personas pierden mucho tiempo encontrando un lugar en donde aparcar su vehículo, incluso suelen ingresar a un parqueadero y al no encontrar espacio se deben dirigir a otro, lo cual ocasiona retrasos en las actividades académicas y laborales. Esto se debe a que no existe un método para conocer el aforo actual de los parqueaderos de la institución. Este proyecto propone desarrollar una plataforma tecnológica que muestre datos sobre la afluencia de vehículos en los sectores de mayor relevancia identificando los parámetros claves que sirvan de ayuda en la toma de decisiones.

1.3. Objetivos

A continuación, se detalla el objetivo general del proyecto integrador, así como también los objetivos específicos que facilitarán el desarrollo del proyecto.

Objetivo General

Diseñar una aplicación móvil que permita monitorear el aforo en sectores específicos empleando cámaras para optimizar el uso de recursos y áreas del campus ESPOL Prosperina.

Objetivos Específicos

1. Analizar la situación actual de los sectores de interés para identificar parámetros de medición y evaluación.
2. Implementar un sistema de identificación de aforo para determinar la cantidad de personas o vehículos que se encuentren en un área específica.
3. Recopilar información histórica del monitoreo de aforo en un servidor de base de datos.

1.4. Marco teórico

El propósito de los sistemas de estacionamiento inteligente es facilitar a los conductores la búsqueda eficiente de lugares de estacionamiento a través de tecnologías de información y comunicación. Aunque la idea inicial se planteó hace una década, en los últimos cinco años se ha observado un creciente interés en los sistemas de estacionamiento inteligente. Esto se debe en gran medida al rápido aumento en el uso de teléfonos inteligentes, que permiten a las personas conectarse a Internet y acceder a información incluso cuando están fuera de casa.

En un estudio se planteó como meta mejorar el proceso de localización de espacios de estacionamiento para vehículos (Rosales, 2016). La investigación se llevó a cabo en el bloque B de la Universidad Politécnica Salesiana y se centró en docentes y personal administrativo de dicho bloque. La muestra consistió en 12 estacionamientos designados para este grupo. El enfoque de la investigación fue de naturaleza experimental con un enfoque tecnológico-deductivo. Los resultados del estudio incluyen el desarrollo de un sistema electrónico, la programación del microcontrolador ATmega328, la implementación en la plataforma de nube Portal Temboo y el diseño de una interfaz web para la integración de aplicaciones. Según las conclusiones del autor, el sistema de estacionamiento inteligente basado en IoT permite ofrecer información en tiempo real a los usuarios acerca de la disponibilidad de espacios para aparcar sus vehículos.

En su investigación de tesis, se diseñó un sistema de acceso de vehículos para las instalaciones de la Pontificia Universidad Católica del Perú (Gomero, 2017). Este estudio se enmarcó en la investigación tecnológica aplicada y se optó por utilizar equipos RFID en el desarrollo del sistema. El control se basó en el microcontrolador ATmega328. Los resultados demostraron que, durante las pruebas RFID, el sistema podía identificar etiquetas a una distancia de 4 cm desde la base del lector. Además, en la prueba final de interfaz y control de acceso, solo el 8 por ciento de las lecturas presentaron errores en el total de vehículos. En última instancia, el autor concluyó que la aplicación de detección de placas vehiculares openALPR tenía deficiencias en relación con el ángulo de enfoque de la cámara, especialmente cuando superaba los 30 grados con respecto al vehículo, y que también existían problemas al reconocer placas que contenían letras como “Y” o “Q”. Por lo tanto, se recomendó que la posición de la cámara no debía exceder un ángulo de 30 grados.

Se propuso la creación de un estacionamiento como un modelo para supervisar y gestionar espacios de estacionamiento disponibles (Aguilar, García y Rivas, 2021). La investigación se centró en la aplicación práctica y utilizó un enfoque cuantitativo. Los resultados de las pruebas señalaron que, en el procesamiento de imágenes, el valor del área máxima dentro del contorno cerrado de una plaza de estacionamiento fue de 8756. La interfaz se diseñó para dispositivos Android y se basó en una base de datos en Python. Se llegó a la conclusión de que el sistema de estacionamiento funcionaba

adecuadamente, siempre y cuando la iluminación fuera la adecuada, ya que la reflexión de la luz podría afectar la precisión de las cámaras IP.

En otro estudio se estableció como meta la creación de un sistema autónomo destinado a generar rutas que orienten al conductor hacia un lugar de estacionamiento disponible en un centro comercial (Romero, 2023). Esta investigación tuvo un enfoque práctico y empleó un enfoque cuantitativo. Para el procesamiento de los datos, se utilizó el software Matlab. Como resultado, se logró diseñar una red de sensores, se seleccionaron los dispositivos electrónicos adecuados y se programaron en el Microcontrolador nodeMCU ESP32. Además, se desarrolló una aplicación para dispositivos Android. El autor concluyó que la interfaz resultó fácil de usar para el usuario, ya que presentaba de forma clara diversas opciones visuales.

1.5. Marco conceptual

Plataforma de monitoreo

La cámara de monitoreo no solo puede vigilar mascotas o niños en casa, pero también puede evitar que extraños entren a la casa. Él hace que el usuario esté seguro de la seguridad del hogar. El propósito del monitoreo es para evitar que alguien invada y destruya.

Cuando ocurre el evento, el sistema de monitoreo puede proporcionar video o imágenes como prueba para garantizar los derechos del usuario e intereses. Un sistema de seguimiento también puede causar problemas psicológicos o disuasión de la persona que fue monitoreada y potencialmente ofrecida. La mayoría de los sistemas de seguridad del hogar se pueden clasificar en sistemas controlados localmente y sistemas controlados remotamente.

Los sistemas controlados localmente utilizan un controlador interno para lograr la seguridad del hogar. Esto permite a los usuarios el uso completo de su sistema de monitoreo desde el interior de su hogar a través de un sistema estacionario o interfaz inalámbrica.

Los sistemas controlados remotamente utilizan una conexión a Internet o integración con una vivienda existente sistema de seguridad para permitir a los usuarios un control total de su sistema(s) desde sus dispositivos móviles o computadoras personales (Bing Ying, 2017).

Servicio de estacionamiento/parqueadero

El servicio de estacionamiento desempeña en la actualidad un papel fundamental al término de cada desplazamiento, especialmente en zonas de gran afluencia como áreas comerciales, industriales, de entretenimiento, entre otras. Esto se debe a la elevada presencia de personas que buscan un lugar seguro para estacionar sus vehículos, y estos estacionamientos pueden ser gestionados tanto por entidades públicas como privadas. Los servicios de estacionamiento tienen como objetivo principal satisfacer las necesidades de los conductores, al proporcionar espacios de estacionamiento para sus vehículos, garantizando al mismo tiempo accesibilidad, seguridad y calidad (Cruz, 2013).

Según la forma en que son gestionados, los estacionamientos se pueden clasificar en dos categorías principales: públicos y privados. Estas categorías definen quiénes son responsables de su administración, planificación y supervisión. Los estacionamientos públicos están diseñados para atender a usuarios que necesitan estacionarse por un período breve o moderado, sin incurrir en un costo económico directo (Instituto Metropolitano de Planificación Urbana, 2018).

Por otro lado, los estacionamientos privados se caracterizan por ofrecer aspectos como accesibilidad, seguridad y calidad en el servicio. Están dirigidos a usuarios que requieren un espacio de estacionamiento en función de la duración de su estancia, a cambio del pago de una tarifa correspondiente (Instituto Metropolitano de Planificación Urbana, 2018).

De acuerdo con su ubicación, los servicios de estacionamiento pueden ser: en espacios públicos, donde la forma más conveniente de estacionar un vehículo es en la carretera, alineado con la acera. Este tipo de estacionamiento se divide en dos categorías: el

primero destinado a servicios de transporte público como paradas de autobuses, colectivos o taxis; y el otro para vehículos privados, como los que se encuentran en áreas residenciales.

El otro tipo se denomina fuera de espacios públicos que son áreas específicas preparadas para el estacionamiento de vehículos, que pueden variar desde terrenos pavimentados hasta estructuras subterráneas o elevadas (Cruz, 2013). Por otro lado, de acuerdo con su funcionalidad se puede dividir en laboral cuando los estacionamientos especialmente diseñados para el personal que trabaja en empresas, compañías, industrias y otros negocios.

El tiempo de estacionamiento suele estar vinculado a las horas de trabajo de los empleados. Otro tipo es el comercial que son los estacionamientos destinados tanto a vehículos de proveedores como a clientes que hacen uso de ellos por períodos de tiempos cortos o moderados. Además, del tipo educativo que son los estacionamientos ubicados en áreas cercanas a instituciones educativas. Y el recreativo que se refiere a aquellos situados en áreas de entretenimiento como parques, teatros, centros acuáticos, zonas turísticas, etc. El período de estacionamiento en estos lugares tiende a ser de duración moderada (Chancusig Sánchez, 2021).

Tipos de aforo de tránsito vehicular

Los aforos de tráfico se refieren a la medición del flujo de vehículos en una infraestructura, área o localidad, con la finalidad de clasificar los vehículos que atraviesan esa zona durante un período de tiempo predeterminado. Esta clasificación implica categorizar los vehículos de acuerdo con las normas establecidas en la normativa NTE INEN 2656, con el propósito de determinar la proporción de cada tipo de vehículo en el flujo general.

El objetivo principal de llevar a cabo este conteo de tráfico es obtener una comprensión del promedio diario anual de tráfico, conocido como TPDA, a partir de los datos recopilados. Es importante destacar que la recopilación de información puede ser continua a lo largo del año o tener una base semanal (NTE INEN 2248, 2016).

Existen diversas modalidades para recopilar datos sobre el volumen de tráfico en los conteos de tránsito vehicular. Estas modalidades son las siguientes: a) aforo manual: Implica el registro de vehículos por medio de un observador (persona) posicionado en un punto específico dentro de la área de estudio, b) aforo combinado: este enfoque permite combinar tanto métodos manuales como automáticos para recopilar información sobre el tráfico, c) aforos automáticos: se obtienen utilizando dispositivos mecánicos que cuentan el flujo de vehículos según características específicas, y d) foro direccional: esta modalidad de conteo registra la mayor cantidad de movimientos posibles en la zona de estudio (Chancusig Sánchez, 2021).

Conteo vehicular

La obtención de datos primarios sobre el tráfico vehicular constituye el elemento fundamental en el proceso de gestión de una red de carreteras y en la creación de estrategias de regulación del tráfico. La evaluación del tráfico implica una evaluación integral de diversas características de la región, en términos de su economía, y el mejoramiento de la movilidad ejerce un impacto directo en la productividad de la región.

La aplicación de las metodologías adecuadas para la recopilación de información es de suma importancia para realizar un diagnóstico preciso del desplazamiento de personas y mercancías, así como para el análisis y desarrollo de futuros sistemas de transporte apropiados, tanto en el ámbito público como en el sector privado (Figueroa, 2016).

Hoy en día, predominan fundamentalmente dos categorías de métodos de conteo de tráfico: conteos manuales y conteos automatizados. Al evaluar cuál de estos enfoques es más apropiado para la recopilación de datos, es esencial determinar en primer lugar la extensión de la información que se busca capturar, es decir, la cantidad, precisión y nivel de detalle requerido en los datos.

Aunque no existen diferencias sustanciales entre una metodología y la otra, la comprensión del volumen de tráfico a contabilizar y las diversas variables que necesitan

ser examinadas establece una relación directa con aspectos económicos, así como la cantidad y la calidad de la información recopilada (Figueroa, 2016).

Diversos tipos de sensores se emplean con regularidad para abordar la tarea de contar vehículos, entre los cuales se incluyen el lazo inductivo, el flujo magnético, el radar de microondas, los sensores ultrasónicos, el radar láser, el lidar, los sensores infrarrojos, las mangueras neumáticas, cámaras digitales y combinaciones de diferentes tecnologías.

A pesar de que los sensores magnéticos, ultrasónicos, infrarrojos y de microondas son los más comúnmente utilizados en el conteo de vehículos, presentan limitaciones cuando se requiere obtener información minuciosa. En los últimos años, las cámaras digitales han entrado en uso para la detección y conteo de vehículos, pero tienen el inconveniente de necesitar un considerable poder de procesamiento para extraer la información necesaria. El proceso general que se sigue para llevar a cabo el conteo implica la instalación de los dispositivos de captura en una ubicación específica y el inicio de la adquisición de datos (Cruz, Lozano, Noriega, Vergara, 2020).

No obstante, es necesario realizar múltiples pruebas antes de determinar la ubicación adecuada para colocar los sensores, por lo que, en algunos casos, se recurre a simulaciones para evaluar cómo enfrentar las condiciones del entorno (Cruz, Lozano, Noriega, Vergara, 2020).

Sistema de parqueo inteligente

Los sistemas de estacionamiento inteligente se presentan como una solución primordial para aliviar la creciente congestión del tráfico. Tanto investigadores como ciudades han comenzado a implementar servicios de estacionamiento inteligente desde diversas perspectivas. Un componente esencial de estos servicios abarca dos aspectos fundamentales: la información y el tráfico. Los conductores suelen recibir información sobre la disponibilidad de espacios de estacionamiento en la vía que desean utilizar, lo que les permite dirigirse a la ubicación deseada y proceder al estacionamiento de manera más eficiente (Rosales, 2016).

Hay algunos informes en la tarea de la detección de espacio de estacionamiento disponible. Se puede categorizar la detección de espacio de estacionamiento en dos grupos de la siguiente manera: basada en ultrasonido y basada en procesamiento de imágenes. El enfoque basado en ultrasonido utiliza una red de sensores ultrasónicos. El sensor ultrasónico se instala en la parte superior de un espacio de estacionamiento utilizando un sensor por espacio. La detección del espacio disponible se realiza mediante la detección de la distancia de reflexión del sensor ultrasónico. La distancia detectada se envía al controlador para gestionar los espacios de estacionamiento disponibles. Este enfoque es muy preciso, pero no es rentable tanto en la instalación como en el mantenimiento (Huang, Liu, Luo, Wang, Yang, 2022).

Sensor de ultrasonido

En esencia, existen dos sistemas activos para medir distancias o proximidad sin necesidad de contacto físico: los sistemas ópticos y los sistemas ultrasónicos. Los sistemas ópticos ofrecen una mayor precisión gracias a su longitud de onda más corta y su menor susceptibilidad a condiciones ambientales, como la presión y la temperatura. Por otro lado, las aplicaciones basadas en la medición del tiempo de vuelo de ultrasonidos son más sencillas y, por lo tanto, menos costosas. Sin embargo, en estos últimos sistemas, presentan una serie de desafíos que requieren atención, como la absorción o atenuación en el medio, el ancho del haz ultrasónico, la presencia de ruido e interferencias, la alta sensibilidad a la temperatura y la humedad, así como la resolución limitada (Navarro, Parra, Ríos, 2004).

La sensorización mediante ultrasonidos se ve influenciada por factores ambientales y las características del objeto reflejante. En el caso de aplicaciones que emplean la técnica de pulso-eco para medir distancias, la precisión del sistema se ve principalmente afectada por las variaciones en la velocidad de la onda ultrasónica debido a la temperatura y la composición del medio de transmisión (Navarro, Parra, Ríos, 2004). Un sensor ultrasonido es una tecnología asequible utilizada para la medición de la distancia hasta un objeto mediante la emisión y recepción de ondas. Estos sensores transmiten ondas de energía en un rango de frecuencia que oscila entre 25 y 50 kHz. Su

funcionamiento implica que el cabezal del sensor emita una onda ultrasónica que luego es reflejada por el objeto.

La distancia se calcula al medir el tiempo que transcurre entre la emisión y la recepción de la onda ultrasónica. El sensor se coloca a una distancia de 11.04 cm desde la base del prototipo y con un ángulo de 30 grados. Para llevar a cabo el conteo en tiempo real, se obtiene un valor analógico del sensor y se interpreta en consecuencia. Además, se debe establecer una condición para distinguir cuando el sensor detecta un objeto y cuando detecta el carril (Han, Li, Liu, Lv, 2015).

Base de datos

Hoy en día, se recopilan cantidades masivas de datos a diario en entornos de redes de sensores ubicuos. Con estos datos disponibles y estructurados de manera elaborada, es más fácil encontrar y acceder a conocimientos y tendencias por medio de la utilización de técnicas de minería de datos. Estos datos son valiosos para respaldar análisis y la toma de decisiones en empresas, por ejemplo.

Por lo general, estos datos existen en bases de datos de varios tipos, que se llamarán bases de datos ubicuas en adelante, y que a menudo pueden estar distribuidas en cualquier lugar. Sin embargo, un problema importante es que una persona que se dedica a la minería de datos utilizando bases de datos ubicuas tendría que invertir mucho tiempo en la selección de la base de datos y la recopilación de datos, lo que sería simplemente un paso preparatorio para las tareas reales de minería de datos (Syoubu, Wada, Watanabe, 2010).

Por otro lado, la independencia de datos se consideró un avance importante, ya que liberó a los programadores de lidiar con los detalles de bajo nivel para acceder a datos (compartidos) en términos de rendimiento y concurrencia correcta. Mientras tanto, la tecnología de hardware y software ha cambiado drásticamente.

En consecuencia, el papel de las bases de datos y la tecnología de bases de datos en el entorno actual debe ser reconsiderado y redefinido. Una base de datos es una plataforma

para que los clientes accedan concurrentemente a datos compartidos. Se necesita definición de datos, manipulación de datos, transacciones en la interfaz. La base de datos, bajo la superficie, realiza la optimización de consultas, garantiza la corrección en el acceso paralelo, la recuperación, la persistencia, el equilibrio de carga, el control de admisión y la disponibilidad (Schek, 2001).

Nube

Dado que el uso de la computación en la nube tiene muchos beneficios en términos de mantenimiento de recursos y ahorro de almacenamiento, las empresas y las personas han comenzado a delegar el control directo de todos o parte de sus recursos a nubes públicas. Sin embargo, esto plantea preocupaciones emergentes sobre la violación de la seguridad de los datos externalizados y el control de acceso a los mismos. La encriptación basada en atributos (ABE) aborda este problema al proporcionar un control de acceso flexible sobre los datos externalizados (Hahn, Hur, Kwon, 2018).

Cuando se habla de un sistema de descifrado externalizado verificable se explica que consta de un propietario de datos, un usuario y un servidor en la nube. El papel de cada entidad se puede describir de la siguiente manera: a) propietario de datos: este es el propietario de los datos que se cargarán en un servidor en la nube de acceso público. Dado un elemento de datos, genera un valor de compromiso utilizando la clave de compromiso. El valor de compromiso se carga junto con el texto cifrado en el servidor en la nube (Hahn, Hur, Kwon, 2018).

Otro elemento es el servidor en la nube: este es un proveedor de servicios de almacenamiento en la nube que almacena los pares texto cifrado-valor de compromiso cargados por varios propietarios de datos. Realiza una descifrado parcial en nombre del usuario y envía el resultado al usuario. Se asume que el servidor en la nube no es de confianza. Tiene interés en los datos externalizados y podría manipular los datos o desobedecer las operaciones dedicadas de descifrado parcial (Ferraz, Icaro, Lima, Sampaio, 2018). Esta suposición es razonable porque el servidor en la nube podría tener un fuerte incentivo financiero para devolver un texto cifrado manipulado en lugar de realizar el descifrado parcial desde cero, si tal comportamiento requiere menos esfuerzo

y es poco probable que sea detectado por el usuario (Hahn, Hur, Kwon, 2018). Por último, el usuario envía una solicitud de datos al servidor en la nube. Al recibir el texto cifrado parcialmente descifrado, realiza el descifrado final y verifica si el servidor en la nube realiza la descifrado parcial correctamente (Hahn, Hur, Kwon, 2018).

Cámara web

El procesamiento de imágenes a partir de una cámara de video es otro enfoque que se puede utilizar para la detección automática de espacios de estacionamiento disponibles. Por ejemplo, H. Ichihashi y otros [2] - [3] aprenden imágenes recortadas de automóviles y construyen el clasificador utilizando un clasificador basado en el método de c-medias difusas (FCM) y ajustan los parámetros utilizando la optimización por enjambre de partículas (PSO). Podemos categorizar este enfoque como detección de espacios de estacionamiento disponibles basada en el reconocimiento.

La detección de espacios de estacionamiento disponibles basada en el reconocimiento intenta construir un clasificador aprendiendo las características de un automóvil y clasificando los automóviles en la región de la imagen de entrada. Este enfoque es preciso pero complejo. Requiere un gran conjunto de datos de automóviles en todas las posibles vistas y condiciones de iluminación para construir el clasificador, lo que lo hace poco flexible para los usuarios (Choeychuen, 2012).

Una cámara web, en términos generales, se refiere a cualquier cámara que genera imágenes accesibles a través de un servidor de Internet y que, por lo general, se conecta a una computadora mediante un puerto USB para transmitir imágenes a la web. En esencia, una cámara web se encuentra enlazada a una computadora, ya sea por conexión directa o inalámbrica, y su función principal consiste en capturar imágenes para su visualización a distancia. No obstante, las capacidades de una cámara web son aprovechadas por usuarios de todo el mundo para diversos propósitos, lo que las convierte en dispositivos versátiles aptos para su uso en prácticamente cualquier entorno (Marker, 2021)

Aplicaciones

Las aplicaciones son programas que permiten acceder a las funciones del sistema. Estas aplicaciones son flexibles y se ajustan al entorno, respondiendo de manera variada a diferentes circunstancias y al tipo de servicio requerido. Pueden ejecutarse tanto en el mismo dispositivo como de manera remota, incluso en dispositivos móviles con recursos de procesamiento limitados, ya que la carga computacional se asigna a los agentes y servicios correspondientes (Tubon, 2020).

El proceso de desarrollo de aplicaciones implica varias fases, y es esencial que el desarrollador siga todas ellas, prestando una atención especial al diseño. De lo contrario, la implementación podría no lograr los resultados deseados. Entre los tipos de aplicaciones se encuentran: las móviles, web, de escritorio, empresariales, redes sociales, juegos, mapas, entretenimiento, salud y bienestar, de viajes y de compras en línea. Sin embargo, en el siguiente apartado se mencionarán las aplicaciones móviles, ya que serán usadas para el desarrollo del presente proyecto (Challiol, Gordillo, Lliteras, 2017).

Una aplicación móvil se refiere a un pequeño programa de software diseñado para abordar una o varias tareas específicas. Tanto desarrolladores individuales como empresas privadas ofrecen aplicaciones de software para dispositivos móviles, algunas de las cuales son gratuitas, mientras que otras requieren un pago, dependiendo de las preferencias y necesidades de sus creadores (Pardo Serna, 2016).

Estas aplicaciones tienen la capacidad de mejorar y personalizar un dispositivo móvil de acuerdo con las preferencias y necesidades de cada usuario, quien tiene la libertad de elegir e instalar las aplicaciones que considere útiles. Además, es fundamental tener en cuenta que el desarrollo de aplicaciones móviles debe centrarse en la adaptación continua y la evolución de las necesidades. Esto implica no solo la incorporación de nuevos requisitos (adaptación) sino también, en muchas ocasiones, un cambio en el objetivo original (Challiol, Gordillo, Lliteras, 2017).

Por otro lado, Morales y Vivar (2022) indicaron que se pueden identificar tres categorías de aplicaciones móviles, entre ellas las nativas, las híbridas y las aplicaciones web. Las aplicaciones nativas se desarrollan utilizando los lenguajes de programación propios del sistema operativo y aprovechan al máximo el hardware de los dispositivos gracias a los paquetes de desarrollo del sistema.

Las aplicaciones híbridas combinan diversas tecnologías de lenguaje del sistema operativo, incorporando elementos web en su interfaz. En cuanto a las aplicaciones web, aunque se asemejan a las aplicaciones nativas, se distinguen por hacer uso exclusivo de la tecnología web.

Python

Fue inventado por los neerlandeses en 1991 utilizando el lenguaje ABC, sin que se confunda con el lenguaje C. La mayoría de las características de ABC se convirtieron en Python, ya que Python es gratuito y sus extensiones se incluyen en las descargas básicas. Python 2.0 fue creado en el año 2000 con algunas características asombrosas, y Python 3.0 se inventó en 2008. La mejor y más compatible versión de Python es la 2.7, que se lanzó en 2015. Puede admitir interfaces de multiprogramación (Mahalakshmi, Pandarachalil, Sendhilkumar, 2015).

Por lo tanto, se puede decir que Python abarca desde la administración del sistema hasta el diseño web, e incluso el manejo de números y es un punto de gran interés en la mente de los investigadores, generando ideas que están mejorando continuamente en comparación con MATLAB y R. Python se destaca por su autenticidad en habilidades de programación y enfoque práctico, ya que puede resolver muchos de los problemas de MATLAB y R, y cuenta con bibliotecas y métodos incorporados para la programación en la investigación científica real. Python se encuentra entre los patrones de diseño orientados a objetos y funcionales. Python utiliza muchas características de lenguaje C, como componentes, bibliotecas iteradas, definiciones de clases, funciones, objetos, etc., que pasan muy fácilmente a través de Python. Tiene una buena capacidad de legibilidad y su sintaxis es fácil de entender para los programadores. Python también fomenta la creación de código reutilizable (Supriya, 2019).

1.6. Estado del arte

Durante el desarrollo de la investigación, se han encontrado artículos científicos relacionados con nuestro el central. Por ejemplo, en 2017 se propuso un sistema de seguimiento de vehículos basado en GPS-GSM que se controlaba a través de una aplicación móvil basada en Google Maps.

Este enfoque se relaciona con el Internet de las cosas (IoT), que es un campo de gran interés en la actualidad y busca la colaboración interdisciplinaria para unificar y facilitar el control y comprensión de diversos aspectos en el mundo. El seguimiento de vehículos es una aplicación específica de IoT en la que se utiliza una antena GPS, un módem GSM, un microcontrolador Atmega y una aplicación móvil para ubicar el vehículo en un mapa y ayudar al usuario a encontrarlo cuando se ha extraviado (Lema Puzhi, 2019).

En una investigación de relevancia internacional, se destaca el trabajo llevado a cabo por Astudillo y Delgado (2012) titulado sistema de localización, monitoreo y control vehicular basado en los protocolos GPS/GSM/GPRS. Su objetivo principal radica en implementar el seguimiento vehicular a través de tecnologías satelitales, con acceso a la información a través de una plataforma en línea.

Esta plataforma consta de una serie de formularios que permiten a los usuarios seleccionar la fecha, hora, ruta y número de placa, lo que facilita las tareas de monitoreo y seguimiento de vehículos. Los datos se transmiten a través de la red GSM/GPRS hacia servidores donde se almacena toda la información relacionada con la ubicación, la empresa y el vehículo en cuestión.

En estas aplicaciones también se hacen uso de Sistemas de Información Geográfica (GIS), que son útiles para delinear áreas geográficas donde se puede localizar el vehículo. Además, se emplean dispositivos de Localización Automática de Vehículos (AVL), que permiten el seguimiento en tiempo real y posterior (Tirabassi, 2011). También es posible calcular la velocidad del vehículo en función de la zona por la que circula, considerando niveles de velocidad bajos, medios y altos, y teniendo en cuenta los límites de velocidad definidos por las autoridades para cada tipo de vía.

La Internet de las Cosas (IoT) ha habilitado la capacidad de monitorear, activar y supervisar dispositivos desde cualquier lugar. En un estudio realizado por Alvear et al. (2017), se diseñó un sistema integrado para la vigilancia a distancia de vehículos. Para lograr esto, se emplearon sensores inalámbricos como Arduino y Raspberry Pi, junto con lenguajes de programación como Java y Javascript. Además, se utilizaron los marcos de trabajo Primefaces y JavaServer Faces, así como la API de Google para la visualización y seguimiento por medio de satélites, entre otros recursos tecnológicos.

Además, la creación de aplicaciones permite llevar a cabo el monitoreo desde la pantalla de un dispositivo móvil con acceso a Internet (Márquez, 2018). En conjunto con esto, la visión por computadora desempeña un papel esencial en la identificación y lectura de matrículas vehiculares, aprovechando el reconocimiento óptico de caracteres, tal como se expone en el estudio de Álvarez (2017) quien desarrolló un sistema que detecta y lee las matrículas de vehículos mediante la visión por computadora, con el respaldo de tecnologías como el algoritmo de reconocimiento óptico de caracteres y la librería de código abierto OpenCV, que se basa en el lenguaje de programación C++.

Para la adquisición y procesamiento de imágenes, se empleó el detector SURF (Speeded Up Robust Features), que se fundamenta en el sistema de transformación de información SIFT (Scale Invariant Feature Transform).

1.7. Marco legal

Ley orgánica de protección de datos personales

Este proyecto se basará en la Ley Orgánica de Protección de Datos Personales (2021) la cual en el Artículo 7 sobre el tratamiento legítimo de los datos de las personas menciona que puede darse bajo condiciones como:

- Por consentimiento del titular para el tratamiento de sus datos personales, para una o varias finalidades específicas.
- Que sea realizado por el responsable del tratamiento en cumplimiento de una obligación legal.

- Que sea realizado por el responsable del tratamiento, por orden judicial, debiendo observarse los principios de la presente ley.
- Que el tratamiento de datos personales se sustente en el cumplimiento de una misión realizada en interés público o en el ejercicio de poderes públicos conferidos al responsable, derivados de una competencia atribuida por una norma con rango de ley, sujeto al cumplimiento de los estándares internacionales de derechos humanos aplicables a la materia, al cumplimiento de los principios de esta ley y a los criterios de legalidad, proporcionalidad y necesidad.

Por otro lado, en el Artículo 8 acerca del consentimiento de estos datos personales se explica que pueden comunicarse o tratarse cuando de manera voluntaria el titular ha otorgado su consentimiento para hacerlo y cuando su voluntad sea expresada de manera:

- Libre, es decir, cuando se encuentre exenta de vicios del consentimiento.
- Específica, en cuanto a la determinación concreta de los medios y fines del tratamiento.
- Informada, de modo que cumpla con el principio de transparencia y efectivice el derecho a la transparencia.
- Inequívoca, de manera que no presente dudas sobre el alcance de la autorización otorgada por el titular. (Ley Orgánica de Protección de Datos Personales, 2021, p.8)

INEN

De acuerdo con la norma INEN 2248 (2016) se especifica que los puestos para los estacionamientos de vehículos se clasifican de la siguiente manera: plazas a 30°, plazas a 45°, plazas a 60°, plazas a 90° y por último las plazas que se ubican en paralelo. Además, de acuerdo con la misma normativa, las plazas deben contar con las siguientes dimensiones, donde a ese el ancho, b la longitud y h el alto.

CAPÍTULO 2

2. METODOLOGÍA

En esta sección se efectúo la ejecución de los procedimientos necesarios para desarrollar una solución que cumpla con los objetivos planteados. Se describe en detalle la recopilación de datos en campo con su respectiva evaluación para conocer de mejor manera los antecedentes de la problemática, así como también, la exposición de los materiales y métodos a utilizar para el desarrollo del Proyecto Integrador. Cabe destacar que, para la implementación de la solución se recurrió a realizar una simulación basado en un ambiente de programación.

2.1. Técnicas de investigación

En el siguiente apartado se muestran las distintas técnicas y procedimientos realizados para alcanzar los objetivos del proyecto.

Levantamiento de la información

Se investiga y recopila información, en distintas fuentes, de los métodos existentes para realizar un control de aforo en parqueaderos. Se verifican distintas tecnologías, programas, aplicaciones y componentes que se requieren para realizar los monitoreos.

Análisis de la información levantada

Con la información obtenida previamente se analiza que tecnologías y materiales son lo más adecuados para el alcance del proyecto, tomando en cuenta la situación que se presenta en distintos escenarios como horas pico generado por el ingreso masivo de vehículos en distintos momentos del día en intervalos de tiempo.

2.2. Descripción del modelo de simulación

En esta sección se detallan los diferentes componentes y aspectos que integraron el modelo de simulación. Para esto, se requirió el uso de herramientas de software especializadas en el desarrollo de aplicaciones que incorporan servicios de transmisión y recepción de datos a través de Internet.

Diseño de la solución

La solución consiste en obtener imágenes en tiempo real de los distintos puntos de interés, los cuales son captados por un sistema de videocámaras que transmiten dichas imágenes a un servidor web que ejecuta un algoritmo complejo desarrollado con el lenguaje de programación Python capaz de detectar la presencia de personas y vehículos dentro del área designada gracias al uso de la tecnología de Visión Artificial.

Este algoritmo se basa en el uso de una librería denominada CV2 especializada en procesamiento de imágenes y vídeo. Así mismo, el algoritmo utiliza la técnica de sustracción de fondo para identificar objetos en movimiento mientras se descarta o elimina el resto de la escena, permitiendo así determinar el número de personas o vehículos con mayor precisión.

En la figura 2.1 podemos observar cómo el programa identifica los patrones que pasan por un área establecida, la misma que nos permite contar el ingreso o salida de vehículos de los sectores de interés.

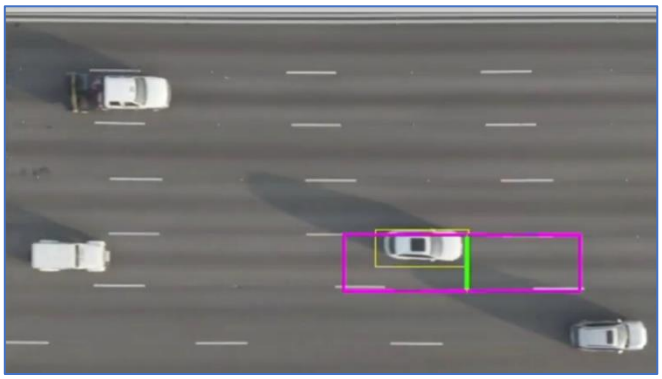


Figura 2.1: Identificador de patrones

El funcionamiento del programa consiste en analizar las imágenes capturadas del vídeo y dividirlos en fragmentos denominados frames y con la ayuda de las librerías podemos determinar la orientación en la que pasa un vehículo. De esta forma se puede determinar si un vehículo ingresa o sale del perímetro controlado. Esto se utiliza para incrementar o disminuir el contador y poder determinar el aforo actual. Podemos apreciar su comportamiento en la figura 2.2.

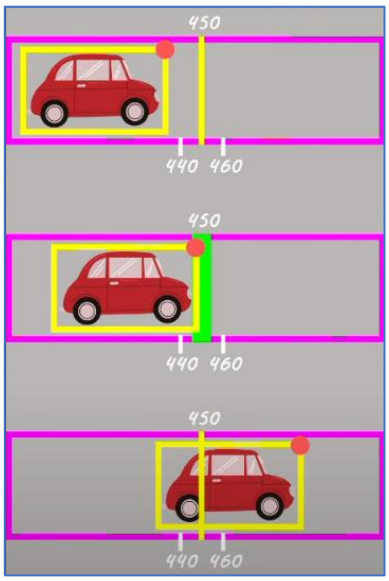


Figura 2.2: Visualizador de frames

Una vez que las imágenes fueron procesadas por el algoritmo, los datos obtenidos son enviados a un servidor de base de datos, mismo que se encuentra alojado en una plataforma en la nube de Google denominado Firebase, en donde se almacena la información para posteriormente ser recuperados por la aplicación móvil a través del uso de una API-REST vía HTTP. La base de datos tiene una estructura no relacional lo cual otorga flexibilidad en el manejo de datos y mayor escalabilidad en la implementación del proyecto.

Por último, se desarrolló una aplicación móvil para dispositivos inteligentes con sistema operativo Android que tiene la capacidad de mostrar los datos procesados en tiempo real en una interfaz intuitiva y sencilla para el usuario. Además, este aplicativo cuenta con la integración de ESPOL Ads que ayuda a la visualización de noticias y actualizaciones de ámbito general dentro del campus ESPOL.

Servidores

Para responder con los requerimientos que la solución conlleva es necesario contar con servidores que estén disponibles para garantizar el correcto funcionamiento y puesta en marcha del proyecto. El servidor web alojará un script que se estará ejecutando en primera instancia para recibir y procesar las imágenes enviadas por la cámara IP y, por consiguiente, realizar el seguimiento del aforo de las áreas de interés. Para el despliegue del servidor de base de datos se usará Firebase Realtime Database mismo que está hospedado en la nube de Google.

Comunicación entre los servidores

A continuación, se describe los procesos de comunicación entre los componentes que forman parte de la solución, así como los servicios propuestos para la implementación de cada uno. La figura 2.3 muestra el flujo de comunicación entre servidores.

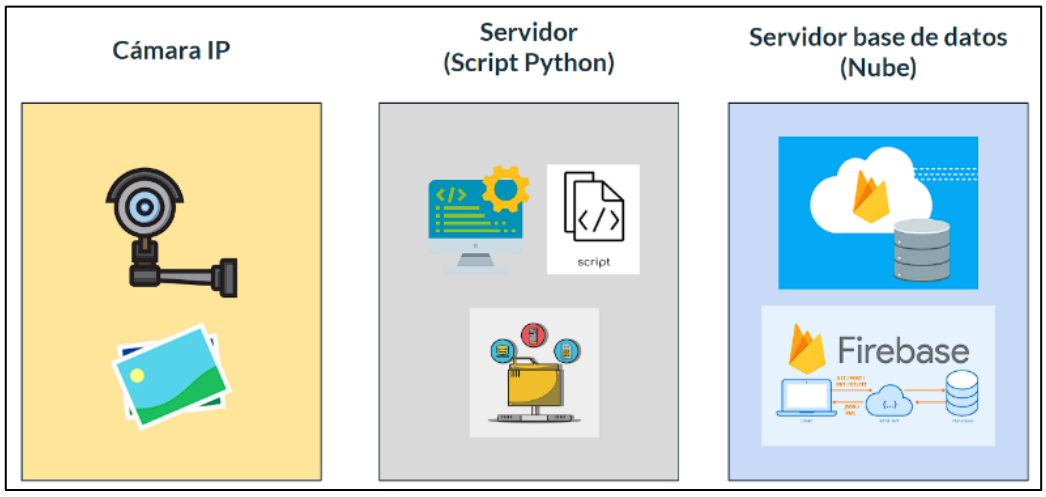


Figura 2.3: Etapas de comunicación entre los componentes

Primero, las cámaras IP captan las imágenes en tiempo real para luego transmitirlas hacia el servidor web donde serán procesadas gracias al algoritmo basado en Visión Artificial, encargado de llevar un contador que señale la cantidad de personas o vehículos entrantes y salientes dentro del entorno de aplicación.

Posteriormente, se realiza una conexión con el servidor de base de datos para así almacenar la información procesada. Por último, la aplicación móvil será capaz de recuperar los datos mediante el uso de una API y así poder mostrarlos en pantalla al usuario final.

Configuración y despliegue del proyecto en Firebase

Para poder crear una instancia de la base de datos en la plataforma de Firebase, se debe contar con una cuenta de Google que ayudará a realizar la configuración, administración y seguimiento del proyecto. Para la configuración de la base de datos primero se debe establecer un nombre al proyecto, en este caso Aforo Espol. Posterior a esto, se procede a vincular nuestro proyecto con la aplicación móvil para el cual se debe registrar el nombre del paquete y el certificado de firma SHA-1 generado por el entorno de desarrollo Android. Este proceso genera unos archivos de configuración para que sean colocados dentro de la carpeta raíz del proyecto Android. Así mismo, una vez vinculado el aplicativo se debe agregar el SDK de Firebase para hacer uso de las funcionalidades de esta plataforma.

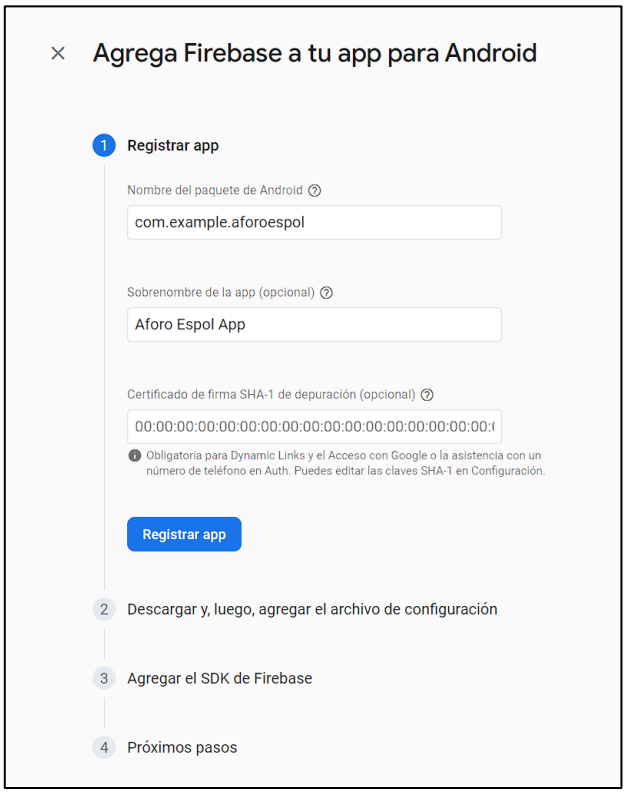


Figura 2.4: Vinculación del aplicativo Android con Firebase

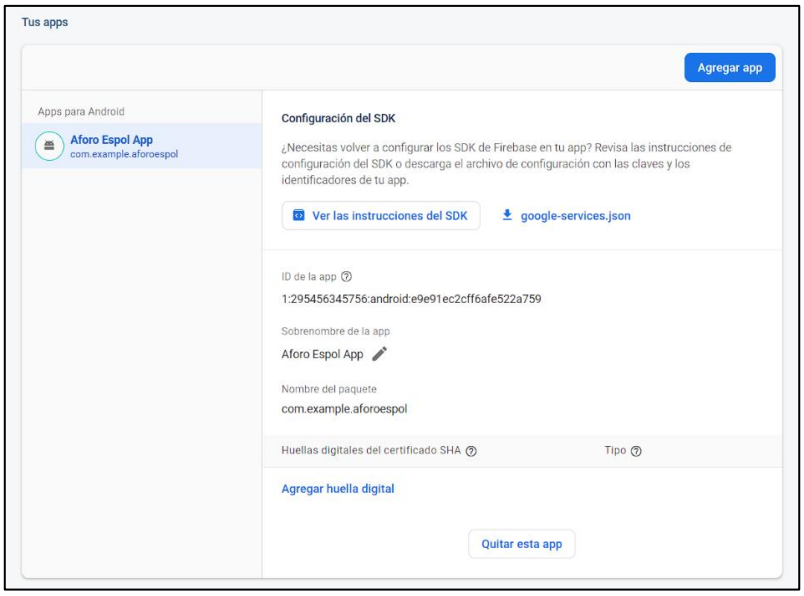


Figura 2.5: Configuración del SDK de Firebase con aplicativo Android

Posterior a la configuración y vinculación del proyecto, se procede a crear una instancia de la base de datos la cual se le configura el parámetro de ubicación y las reglas de seguridad. Finalmente, se empieza a estructurar la base de datos de acuerdo con las necesidades del proyecto.

Diseño de la aplicación móvil

Los dispositivos móviles hoy en día forman parte de la vida cotidiana de las personas siendo Android el sistema operativo más usado a nivel mundial con una cuota de mercado del 84% aproximadamente en el año 2020 según datos de la consultora de tecnología IDC (Mena Roa, 2021). Bajo este contexto, se decidió desarrollar una aplicación móvil orientado a usuarios de dispositivos Android cuyo proceso de diseño se explica a continuación.

Se escogió Android Studio como entorno de desarrollo en su versión 2022.2.1 (Flamingo) con la utilización de JAVA como lenguaje de programación principal. Para la obtención de datos se requiere el uso de las librerías de Firebase para la integración con las bases de datos que ayudarán al despliegue de gráficos y tablas y, de esta manera, poder visualizarlos al ingresar a la aplicación.

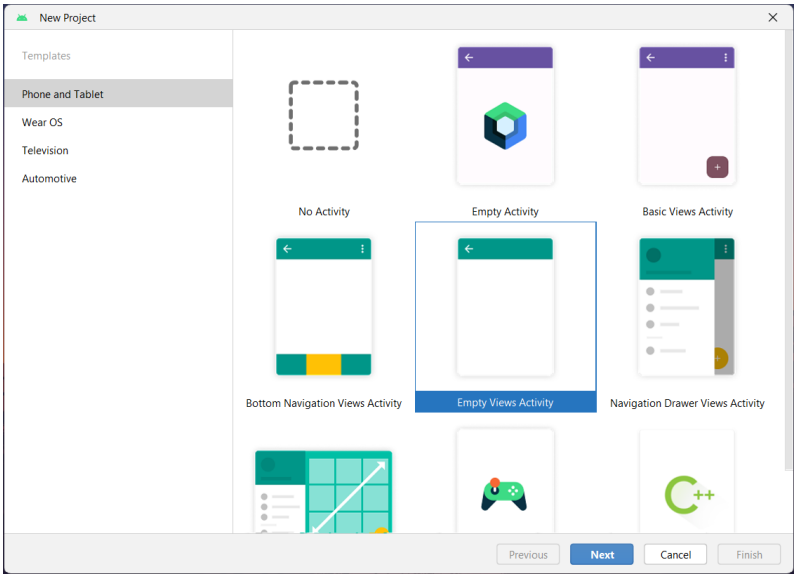


Figura 2.6: Configuración del proyecto en Android Studio

Para configurar el ambiente de desarrollo, se crea un proyecto con una actividad en blanco como se describe en la figura 2.6. Posteriormente, se establece el nombre de la aplicación, el nombre del paquete de Android, el lugar donde se guardará localmente el proyecto, el lenguaje de programación y la versión mínima de Android que deben tener los teléfonos inteligentes de los usuarios para ejecutar el aplicativo. La versión mínima configurada para el aplicativo es Android versión 6.0 lo cual funcionará para el 97,7% de los dispositivos.

Siguiendo los requerimientos del proyecto para determinar el aforo de los parqueaderos de forma precisa se desarrolla la aplicación móvil con una interfaz amigable con el usuario usando los estándares básicos de programación, incluida la programación orientada a objetos de JAVA. La aplicación móvil cuenta con una pantalla principal que contiene un botón para acceder a los diferentes parqueaderos de la ESPOL clasificados de acuerdo con la ubicación de estos. En esta pantalla se muestran detalles del aforo de cada parqueadero como la cantidad de espacios disponibles, ocupados y el aforo total, mismo que se puede visualizar en la figura 2.7.

Figura 2.7: Panel informativo con detalles del aforo de un parqueadero

Al pulsar en el botón para visualizar los detalles de un parqueadero se despliega una pantalla en donde se muestra en tiempo real el estado del aforo que cuenta con un panel informativo basado en un código de colores para identificar si un parqueadero tiene espacios disponibles para aparcar. Este código de colores cambia programáticamente y se compone del color verde, cuando el aforo actual está en un rango del 0% al 45% de la capacidad máxima, el color amarillo cuando está en un rango del 45% al 85% de la capacidad máxima, y el color rojo cuando se encuentra en el rango del 85% al 100% de la capacidad máxima del parqueadero. Se puede observar estas condiciones en el siguiente extracto de código en la figura 2.8.

```
private void updateLinearLayoutContador(Integer aforo_actual) {
    if (aforo_actual >= 0 && aforo_actual < 45 ) {
        linearLayoutContador.setBackgroundColor(Color.parseColor( colorString: "#64D168"));
        img_check.setImageResource(R.drawable.icon_check);
    } else if (aforo_actual >= 45 && aforo_actual < 85) {
        linearLayoutContador.setBackgroundColor(Color.parseColor( colorString: "#FFEFD700"));
        img_check.setImageResource(R.drawable.icon_advertencia);
    } else if (aforo_actual >= 85 && aforo_actual <= 100) {
        linearLayoutContador.setBackgroundColor(Color.parseColor( colorString: "#FFFA695E"));
        img_check.setImageResource(R.drawable.icon_x);
    }
}
```

Figura 2.8: Configuración del proyecto en Android Studio

CAPÍTULO 3

3. Resultados

En esta sección se presentará los resultados obtenidos, tanto del algoritmo de reconocimiento como de la aplicación que muestra el aforo de los puntos de interés. Se describirá paso a paso como se obtienen los resultados y cuál es la finalidad de estos.

Luego de desarrollar el script que permite la identificación de patrones para realizar el conteo de vehículos se toman videos de pruebas para validar el correcto funcionamiento y se obtienen las siguientes evidencias.

Prueba 1

En esta prueba se realiza el conteo en un solo sentido de 3 carriles mismas que son ilustradas en la figura 3.1 y figura 3.2, estos datos representan el ingreso de vehículos en un parqueadero.

Figura 3.1: Video 1 a resolución 720p

Figura 3.2: Video 1 a resolución 480p

Prueba 2

En esta prueba se realiza el conteo en un solo sentido de 2 carriles mismas que son ilustradas en la figura 3.2 y figura 3.3, estos datos representan la salida de vehículos en un parqueadero.

Figura 3.3: Video 2 a resolución 720p

Figura 3.4: Video 2 a resolución 480p

Prueba 3

Para esta prueba se toma en cuenta el carril izquierdo como si fuera el ingreso de vehículos al parqueadero y el carril derecho como la salida de vehículos.

Se puede observar en la figura 3.5 y figura 3.6 que se dispone de 2 carriles para cada caso esto se debe a que existen parqueaderos con múltiples entradas por lo que se requiere realizar conteos en cada una de estas para poder determinar el aforo total.

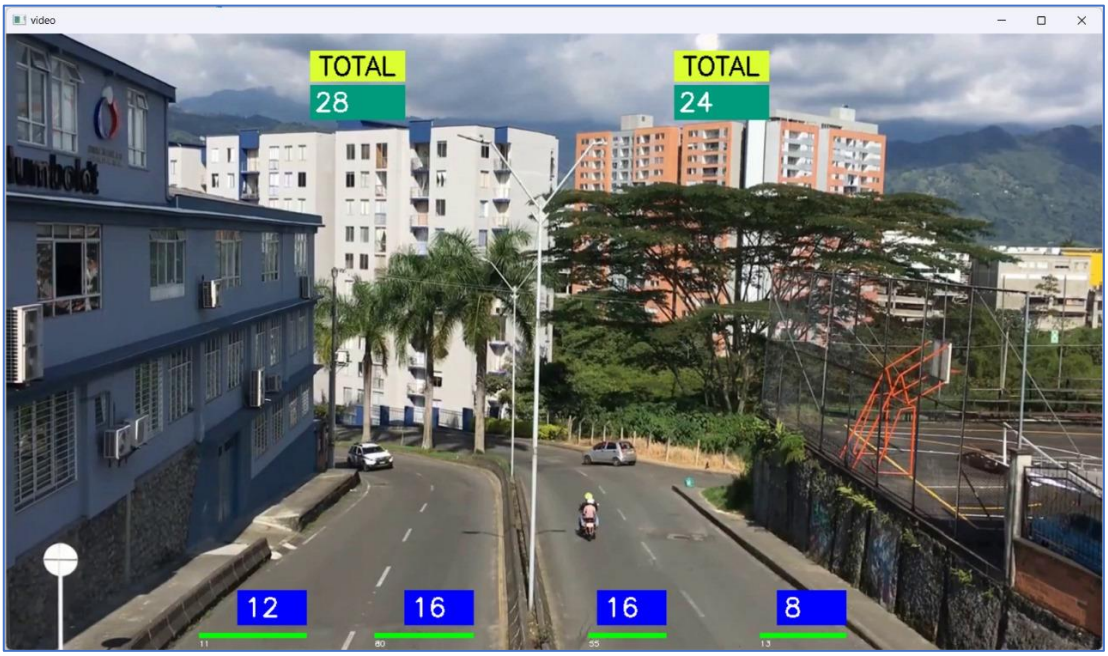


Figura 3.5: Video 3 a resolución 720p

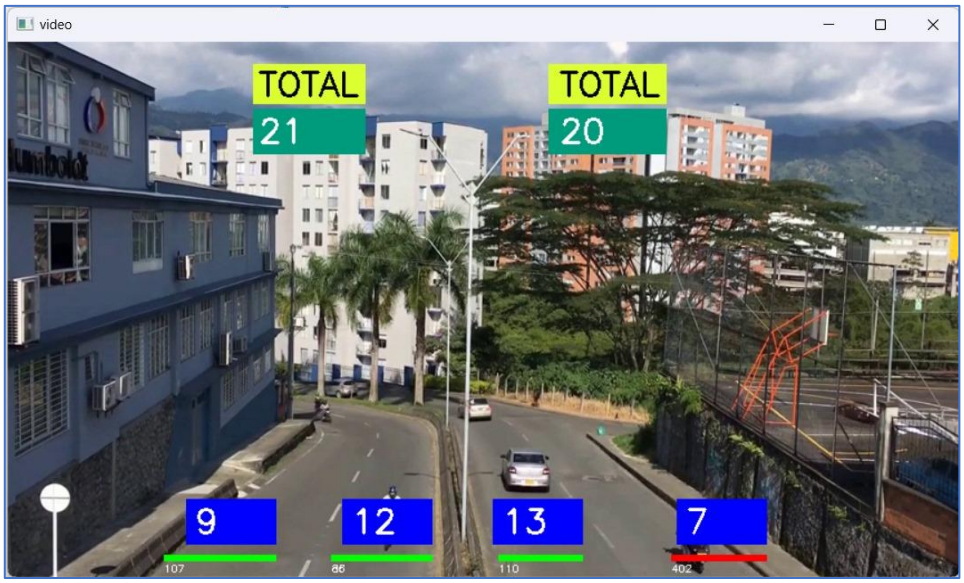


Figura 3.6: Video 3 a resolución 480p

Los resultados en cada prueba fueron satisfactorios, obteniendo así el correcto funcionamiento del programa y conteo de vehículos.

Esta información es enviada al servidor de base de datos en la nube como se puede apreciar en la figura 3.7 para posteriormente ser consultada desde la aplicación móvil.

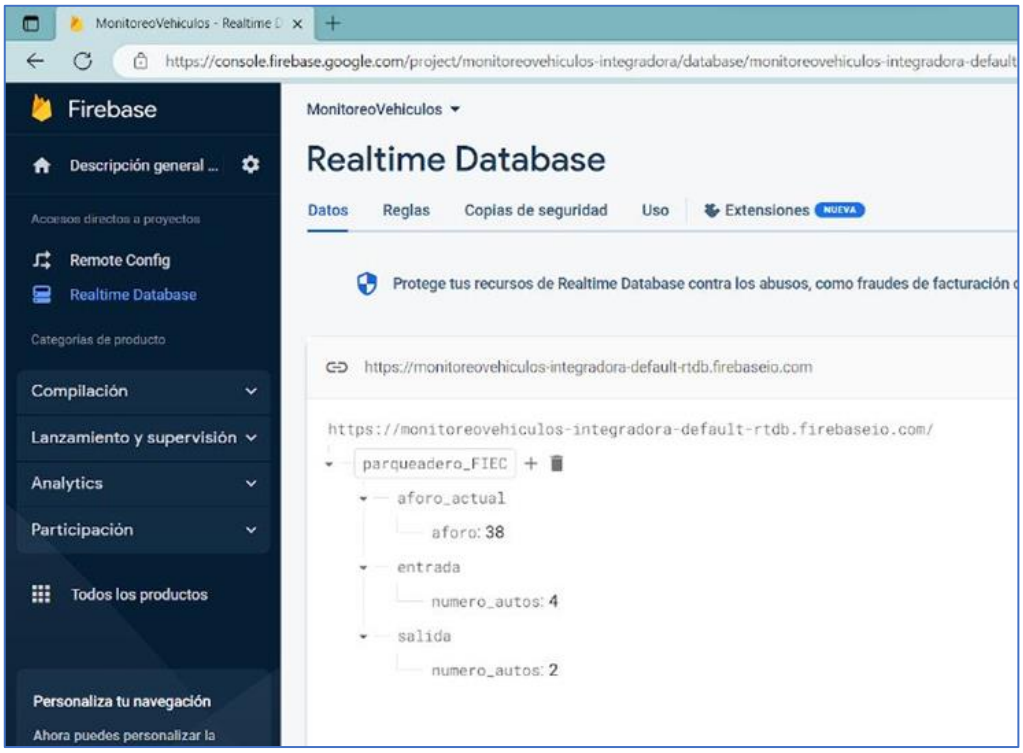


Figura 3.7: Base de datos en la nube

En la aplicación se tienen diferentes panales interactivos y amigables con el usuario. En la figura 3.8 se tiene el panel principal al ingresar a la aplicación.

Figura 3.8: Panel principal

En la figura 3.9 se aprecia un panel que tiene los distintos parqueaderos en los que se realiza el monitoreo del aforo.

Figura 3.9: Panel de parqueaderos

Finalmente, en la figura 3.10 se presenta el panel que muestra el aforo del parqueadero seleccionado mostrando la cantidad de sitios ocupados, se le agregaron colores que representan el porcentaje de ocupación siendo menor al 45% un tono verde claro, entre 45% y menor a 85% un tono amarillo y superior a 85% un tono rojo, también dispone de un diagrama pastel con el porcentaje de aforo y finalmente la integración con Espol Ads.

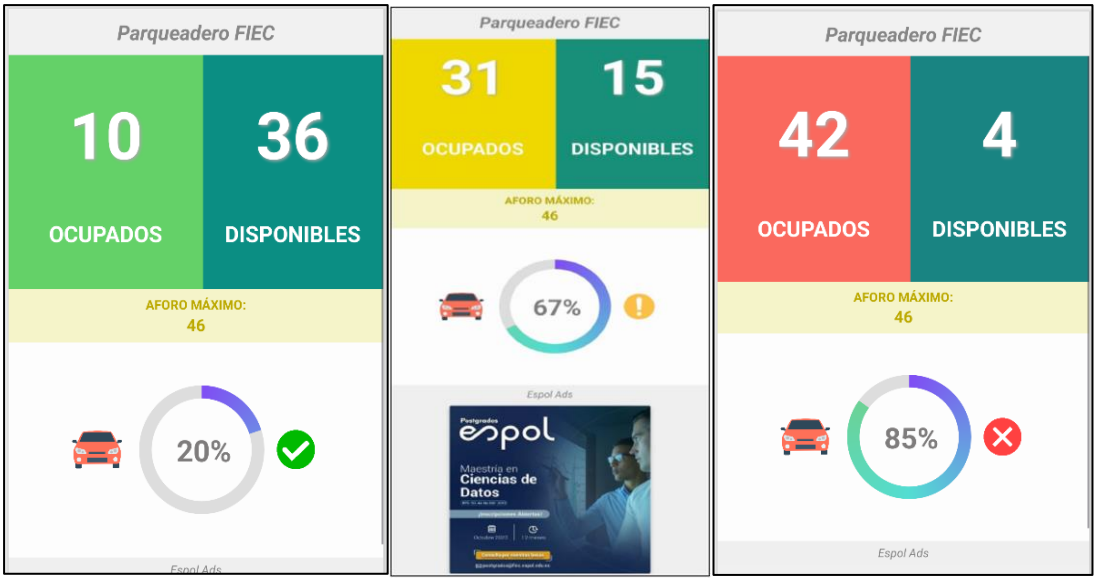


Figura 3.10: Panel de parqueaderos

CAPÍTULO 4

4. Conclusiones y recomendaciones

En esta sección se presentan las conclusiones y recomendaciones obtenidas del proyecto desarrollado.

4.1. Conclusiones

El diseño de la aplicación móvil del proyecto, permite de forma intuitiva la visualización de la disponibilidad de aforo en los diferentes parqueaderos de la institución educativa. Por medio del uso de barras de colores, representa los espacios disponibles con los colores asociados a un semáforo, siendo el color verde de alta disponibilidad, amarillo de media disponibilidad y rojo de poca disponibilidad. Además, se analizó el problema de las personas daltónicas que tienen dificultades en la visualización de los colores, se utilizan unas gráficas de porcentaje mostrando el mismo resultado.

En base a las pruebas realizadas (prueba 1, 2 y 3) descritas anteriormente, se determina que la resolución de video de las cámaras no afecta al algoritmo del contador de vehículos. Por consecuente, el software identifica correctamente los patrones de movimiento de los vehículos para realizar el monitoreo de aforo en los distintos videos de resoluciones de 720p y 480p, los cuales se visualizan en la aplicación móvil.

El uso de la base de datos no relacional Firebase permite realizar consultas en tiempo real de una forma más rápida y precisa, lo cual satisface a los objetivos de este proyecto, permitiendo determinar el aforo de los parqueaderos con datos actualizados al momento que son recopilados directamente desde un dispositivo móvil con conexión a internet para poder acceder a la aplicación desde cualquier lugar.

Los datos recopilados de forma histórica permiten realizar un análisis cuantitativo y evaluar la situación actual para conocer a profundidad los requerimientos y necesidades de cada sector y de esta manera agilizar la toma de decisiones, por ejemplo, ampliar los parqueos existentes según la congestión que se presenta por la alta demanda de vehículos.

4.2. Recomendaciones

En futuros trabajos se puede considerar en un sistema de redundancia para mantener la autonomía del proyecto. Esto implica que se debe considerar otro servidor en caso de que falle el primero y no se vea afectado el monitoreo, de igual forma se debe tomar en cuenta un sistema de suministro energético en caso de que se sufran apagones.

Si se desea considerar el proyecto a gran escala, se debe tener en cuenta el rendimiento del servidor, por lo que podría ser necesario conseguir un procesador más robusto y el aumento de memoria RAM, ya que cada cámara implementada requiere que se ejecute un script.

Para mejorar el rendimiento del algoritmo y aumentar la precisión del contador de vehículos es necesario verificar los parámetros de configuración, como la sensibilidad de lectura de puntos en la sustracción de fondo.

Para garantizar la máxima compatibilidad de la aplicación móvil en cualquier dispositivo inteligente independientemente del sistema operativo, el desarrollo de este debe ser en un ambiente de desarrollo híbrido usando un framework como Flutter.

BIBLIOGRAFÍA

Instituto Metropolitano de Planificación Urbana. (2018). Visión de Quito 2040 y su Nuevo Modelo de Ciudad. Instituto Metropolitano de Planificación Urbana.

Aguilar, J., Garcia, K., & Rivas, W. (2021). Visión artificial para control de estacionamiento vehicular. Polo del conocimiento, 6(9), 2134-2154.

Álvarez, M. (2017). Análisis, diseño e implementación de un sistema de control de ingreso de vehículos basado en visión artificial y reconocimiento de placas en el parqueadero de la Universidad Politécnica Salesiana - Sede Cuenca. Universidad Politécnica Salesiana.

Alvear, V., Peluffo, D., Pijal, J., & Rosero, P. (2017). Internet de las Cosas y Visión Artificial, Funcionamiento y Aplicaciones: Revisión de Literatura. Enfoque UTE, 1, 244-256.

Astudillo, J., & Delgado, E. (2012). Sistema de localización, monitoreo y control vehicular basado en los protocolos GPS/GSM/GPRS. 15° Concurso de Trabajos Estudiantiles.

Bing, C., & Ying, Y. (2017). An Intelligent Embedded Cloud Monitoring System Design. International Automatic Control Conference (CACS), 1-4.

Challiol, C., Gordillo, S., & Lliteras, A. (2017). Diseño de Aplicaciones Móviles basadas en Posicionamiento: un Framework Conceptua. XXIII Congreso Argentino de Ciencias de la Computación, (págs. 682-691).

Chancusig, D., & Sánchez, J. (2021). Estudio de factibilidad para la concesión del servicio de parqueadero del aeropuerto internacional Cotopaxi, provincia de Cotopaxi. Escuela Superior Politécnica de Chimborazo.

Choeychuen, K. (2012). Available car parking space detection from webcam by using adaptive mixing features. Rajamangala University of Technology Rattanakosin.

Cruz, C. (2013). Proyecto de factibilidad para la creación de un parqueadero público en el sector centro de la ciudad de Quito. Universidad Politécnica Salesiana.

Cruz, V., Lozano, J., Noriega, S., & Vergara, O. (2020). Evaluación del desempeño de sensores infrarrojo, ultrasónico y visión para el conteo de vehículos. Research in Computing Science, 149(11), 159-169.

El Universo (2023). Estudiar en la Espol, el sueño de muchos jóvenes.

ESPOL (2023), Espol mejor universidad pública de Ecuador en QS Latin America University Rankings 2024.

Ferraz, F., Icaro, F., Lima, W., & Sampaio, C. (2018). A Disturbing Question: What is the Economical Impact of Cloud Computing? International Conference on Cloud Computing, 853-857.

Figueroa, K. (2016). Aplicación de la tecnología de identificación por radio frecuencia en estudios de tránsito y transporte – parte 1. Universidad Nacional de Colombia.

Gomero, L. (2017). Diseño de un sistema de acceso vehicular a la PUCP basado en tecnología RFID y detección de placas vehiculares. Pontificia Universidad Católica del Perú.

Hahn, C., Hur, J., & Kwon, H. (2018). Toward Trustworthy Delegation: Verifiable Outsourced Decryption with Tamper-Resistance in Public Cloud Storage. International Conference on Cloud Computing, 920-924.

Han, J., Li, B., Liu, J., & Lv, H. (2015). An ultrasonic sensor system based on a two-dimensional state method for highway vehicle violation detection applications. Sensors, 9000-90021.

Huang, C., Liu, Z., Luo, Y., Wang, Y., & Yang, S. (2022). Visual Detection and Image Processing of Parking Space Based on Deep Learning. Research Progress on Intelligent Electric Vehicles, 22(17).

Lema, P., & Puzhi, A. (2019). propuesta de emprendimiento para el servicio de rastreo satelital y protección de menores de edad en tiempo real, mediante dispositivos gps/gprs, a través de la creación de un sistema multiplataforma (web, ios y android) alojado en la nube y aplicación de. Universidad Politécnica Salesiana.

Ley orgánica de protección de datos personales. (2021). Ley orgánica de protección de datos personales. Asamblea Nacional.

Mahalakshmi, G., Pandarachalil, R., & Sendhilkumar, S. (2015). Twitter sentiment analysis for large-scale data: an unsupervised approach. Cognitive computation, 7(2), 254-262.

Marker, G. (2021). Que es una webcam: Cámara de computadora. Obtenido de https://www.tecnologia-informatica.com/camara-web-comprar-webcam/

Márquez, L. (2018). Diseño e implementación de un software de reconocimiento de placas vehiculares en tiempo real. Universidad Nacional Abierta y a Distancia.

Morales, D., & Vivar, F. (2022). Proyecto para el desarrollo de una aplicación móvil para mejorar la productividad y el servicio de pequeños distribuidores de gas doméstico en Cuenca. Universidad Politécnica Salesiana.

Navarro, D., Parra, H., & Ríos, L. (2004). Sensores de ultrasonido usados en robótica móvil para la medición de distancias. Scientia Et Technica, 35-40.

NTE INEN 2248. (2016). Accesibilidad de las personas al medio físico. Estacionamientos. NTE INEN 2248.

Pardo, C., & Serna, S. (2016). Diseño de Interfaces en Aplicaciones Móviles. RA-MA.

Romero, K. (2023). Diseño de un sistema electrónico para el control de ingreso y salida vehicular en un estacionamiento en la Universidad Nacional José Faustino Sánchez Carrión, 2022. Universidad Nacional José Faustino Sánchez Carrión.

Rosales, L. (2016). Diseño e implementación de un parqueo inteligente utilizando arduino yun basado en internet de las cosas (IOT). Universidad Poltiécnica Salesiana.

Schek, H. (2001). Evolution of Database Technology: Hyperdatabases. Swiss Federal Institute of Technology, 139-142.

Supriya, A. (2019). A Survey: How Python Pitches in IT-World. International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (Com-IT-Con), (págs. 248-252).

Syoubu, K., Wada, Y., & Watanabe, Y. (2010). Virtual Database Technology for Distributed Database. WAINA, 214-220.

Tirabassi, A. (2011). Monitoreo satelital de vehículos mediante una aplicación web. Universidad Naciona de La Plata.

Tubon, G. (2020). Aplicación móvil con Georreferenciación para gestión de pedidos a domicilio de un local de comida. Pontifica Universidad Catolica del Ecuador.

APÉNDICE

Tabla de costos de proyecto

Para determinar el costo del proyecto se toma en cuenta el salario de cada colaborador, trabajando 4 horas diarias durante 5 días de la semana por un periodo de 5 meses con sueldo estimado de $300 por mes, aportaciones al IESS por parte del empleador al 11.15%, finalmente, materiales de costo único como las cámaras y el servidor.

Tabla - Costos del proyecto

Nómina			
Personal	**Meses**	**Salario por mes**	**Total**
Andrés Contreras Calle	5	$ 300,00	$1.500,00
Alfonso Enríquez Salguero	5	$ 300,00	$1.500,00
IESS (colaborador 1)	5	$ 33,45	$ 167,25
IESS (colaborador 2)	5	$ 33,45	$ 167,25
Materiales			
Ítems	**Cantidad**	**Costo**	**Total**
Cámaras	12	$ 90,00	$1.080,00
Servidor	1	$ 2.000,00	$2.000,00
Costo Total			
Total			$6.414,50

Código utilizado para la detección y conteo de vehículos (Python)

```
import cv2
import numpy as np

# Puntos/Coordenadas para video 720p [VIDEO 1]

# area1_pts = [225, 700, 125, 8]
# area2_pts = [430, 700, 115, 8]
# area3_pts = [680, 700, 90, 8]
# area4_pts = [880, 700, 100, 8]

# Puntos/Coordenadas para video 480p [VIDEO 1]

# area1_pts = [140, 460, 100, 8]
# area2_pts = [290, 460, 90, 8]
# area3_pts = [440, 460, 75, 8]
# area4_pts = [595, 460, 85, 8]

# Puntos/Coordenadas para video 720p [VIDEO 2]

area1_pts = [265, 680, 140, 8]
area2_pts = [500, 680, 150, 3]
area3_pts = [745, 680, 145, 10]
```

```
# Puntos/Coordenadas para video 480p [VIDEO 2]

# area1_pts = [170, 460, 100, 8]
# area2_pts = [330, 460, 105, 8]
# area3_pts = [480, 460, 105, 8]

# Puntos/Coordenadas para video 720p [VIDEO 3]

# area1_pts = [640, 680, 150, 8]
# area2_pts = [1120, 680, 140, 8]

# Puntos/Coordenadas para video 480p [VIDEO 3]

# area1_pts = [470, 460, 100, 8]
# area2_pts = [730, 460, 105, 8]

#Comentar las coordenadas x4,y4,w4,h4 para el Video 2
#Comentar las coordenadas x3,y3,w3,h3 para el Video 3

x1,y1,w1,h1 = area1_pts
x2,y2,w2,h2 = area2_pts
x3,y3,w3,h3 = area3_pts
# x4,y4,w4,h4 = area4_pts

libre1 = False
libre2 = False
libre3 = False
libre4 = False

num_detect1 = 0
num_detect2 = 0
num_detect3 = 0
num_detect4 = 0

num_total1 = 0
num_total2 = 0

video = cv2.VideoCapture('Seguimiento de Objetos con Python_02_720p.mp4')

while True:
    # ////////////////////////////////////////////////////////////
    #              AJUSTES Y TRANSFORMACIONES DE VIDEO
    # ////////////////////////////////////////////////////////////

    check, frame = video.read()
    frameGris = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
    frameTh = cv2.adaptiveThreshold(frameGris, 254,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, 11, 12)
    # frameBlur = cv2.medianBlur(frameTh, 5)
    kernel = np.ones((8,8), np.uint8)
    frameDil = cv2.dilate(frameTh, kernel, iterations=2)

    # ////////////////////////////////////////////////////////////
    #                              AREA 1
    # ////////////////////////////////////////////////////////////

    recorte1 = frameDil[y1:y1+h1, x1:x1+w1]
    cantPtsBlancos1 = cv2.countNonZero(recorte1)
    cv2.putText(frame, str(cantPtsBlancos1), (x1,y1+15),
cv2.FONT_HERSHEY_SIMPLEX, 0.3, (255,255,255), 1)
```

```
    umbral_sens_1 = 910

    #print("Puntos Blancos 1", cantPtsBlancos1)

    if cantPtsBlancos1 > umbral_sens_1 and libre1 is True:
        num_detect1 += 1
        num_total1 += 1
    if cantPtsBlancos1 < umbral_sens_1:
        libre1 = True
        cv2.rectangle(frame, (x1, y1), (x1 + w1, y1 + h1-3), (0, 255, 0), -1)
# Codigo BGR -> Color verde
    else:
        libre1 = False
        cv2.rectangle(frame, (x1, y1), (x1 + w1, y1 + h1-3), (0, 0, 255), -1)
# Color rojo

    cv2.rectangle(frame, (x1+w1-80,y1-50), (x1+w1,y1-10), (255,0,0), -1)
    cv2.putText(frame, str(num_detect1), (x1+w1-72,y1-20),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2)

    # ////////////////////////////////////////////////////////////
    #                             AREA 2
    # ////////////////////////////////////////////////////////////

    recorte2 = frameDil[y2:y2+h2, x2:x2+w2]
    cantPtsBlancos2 = cv2.countNonZero(recorte2)
    cv2.putText(frame, str(cantPtsBlancos2), (x2,y2+15),
cv2.FONT_HERSHEY_SIMPLEX, 0.3, (255,255,255), 1)

    umbral_sens_2 = 430

    #print("Puntos Blancos 2", cantPtsBlancos2)

    ###########################################
    # Codigo de prueba para mejorar la sensibilidad del sistema para el Video
2

    if cantPtsBlancos2 < 85:
        cantPtsBlancos2 = umbral_sens_2+50

    ###########################################

    if cantPtsBlancos2 > umbral_sens_2 and libre2 is True:
        num_detect2 += 1
        num_total1 += 1
    if cantPtsBlancos2 < umbral_sens_2:
        libre2 = True
        cv2.rectangle(frame, (x2, y2), (x2 + w2, y2 + h2+2), (0, 255, 0), -1)
    else:
        libre2 = False
        cv2.rectangle(frame, (x2, y2), (x2 + w2, y2 + h2+2), (0, 0, 255), -1)

    cv2.rectangle(frame, (x2+w2-80,y1-50),(x2+w2,y2-10),(255,0,0),-1)
    cv2.putText(frame, str(num_detect2), (x2+w2-72,y2-20),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2)

    # ////////////////////////////////////////////////////////////
    #                   CUADRO TOTAL AUTOS (IZQ)
    # ////////////////////////////////////////////////////////////
```

```
    # Cuadro para video 720p
    cv2.rectangle(frame, (x1+130,20), (x2+w2-80,55), (50,255,220),-1)
    cv2.putText(frame, "TOTAL", (x1+140,48), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0,0,0), 2)
    cv2.rectangle(frame, (x1+130,60), (x2+w2-80,100), (125,155,0),-1)
    cv2.putText(frame, str(num_total1), (x1+135,90),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2)

    # Cuadro para video 480p
    # cv2.rectangle(frame, (x1+w1-20,20), (x2+w2-60,55), (50,255,220),-1)
    # cv2.putText(frame, "TOTAL", (x1+w1-15,48), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0,0,0), 2)
    # cv2.rectangle(frame, (x1+w1-20,60), (x2+w2-60,100), (125,155,0),-1)
    # cv2.putText(frame, str(num_total1), (x1+w1-15,90),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2)

    # ////////////////////////////////////////////////////////////
    #                              AREA 3
    # ////////////////////////////////////////////////////////////

    # Se usa num_total2 para AREA 3 y AREA 4 en Video 1 (Carriles de la
derecha)
    # Se usa num_total1 para AREA 3 en Video 2

    # Comentar AREA 3 para el VIDEO 3

    recorte3 = frameDil[y3:y3+h3, x3:x3+w3]
    cantPtsBlancos3 = cv2.countNonZero(recorte3)
    cv2.putText(frame, str(cantPtsBlancos3), (x3,y3+15),
cv2.FONT_HERSHEY_SIMPLEX, 0.3, (255,255,255), 1)

    umbral_sens_3 = 600

    print("Puntos Blancos 3", cantPtsBlancos3)

    if cantPtsBlancos3 < umbral_sens_3 and libre3 is True:
        num_detect3 += 1
        num_total1 += 1
    if cantPtsBlancos3 > umbral_sens_3:
        libre3 = True
        cv2.rectangle(frame, (x3, y3), (x3 + w3, y3 + h3-5), (0, 255, 0), -1)
    else:
        libre3 = False
        cv2.rectangle(frame, (x3, y3), (x3 + w3, y3 + h3-5), (0, 0, 255), -1)

    cv2.rectangle(frame, (x3+w3-80,y1-50), (x3+w3,y3-10), (255,0,0), -1)
    cv2.putText(frame, str(num_detect3), (x3+w3-72,y3-20),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2)

    # ////////////////////////////////////////////////////////////
    #                              AREA 4
    # ////////////////////////////////////////////////////////////

    # Comentar AREA 4 para el VIDEO 2 y VIDEO 3

    # recorte4 = frameDil[y4:y4+h4,x4:x4+w4]
    # cantPtsBlancos4 = cv2.countNonZero(recorte4)
    # cv2.putText(frame, str(cantPtsBlancos4), (x4,y4+15),
cv2.FONT_HERSHEY_SIMPLEX, 0.3, (255,255,255), 1)
    #
    # umbral_sens_4 = 250
```

```
    #
    # if cantPtsBlancos4 > umbral_sens_4 and libre4 is True:
    #     num_detect4 += 1
    #     num_total2 += 1
    # if cantPtsBlancos4 < umbral_sens_4:
    #     libre4 = True
    #     cv2.rectangle(frame, (x4,y4), (x4+w4,y4+h4-3), (0,255,0), -1)
    # else:
    #     libre4 = False
    #     cv2.rectangle(frame, (x4,y4), (x4+w4,y4+h4-3), (0,0,255), -1)
    #
    # cv2.rectangle(frame, (x4+w4-80,y1-50), (x4+w4,y4-10), (255,0,0), -1)
    # cv2.putText(frame, str(num_detect4), (x4+w4-72,y4-20),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2)

    # ////////////////////////////////////////////////////////////
    #                   CUADRO TOTAL AUTOS (DER)
    # ////////////////////////////////////////////////////////////

    # Comentar CUADRO TOTAL para el VIDEO 2

    # Cuadro para video 720p
    # cv2.rectangle(frame, (x3+100,20), (x4+w4-90,55), (50,255,220), -1)
    # cv2.putText(frame, "TOTAL", (x3+110,48), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0,0,0), 2)
    # cv2.rectangle(frame, (x3+100,60), (x4+w4-90, 100), (125,155,0), -1)
    # cv2.putText(frame, str(num_total2), (x3+105,90),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2)

    # Cuadro para video 480p
    # cv2.rectangle(frame, (x3+w3-30,20), (x4+w4-90,55), (50,255,220), -1)
    # cv2.putText(frame, "TOTAL", (x3+w3-20,48), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0,0,0), 2)
    # cv2.rectangle(frame, (x3+w3-30,60), (x4+w4-90,100), (125,155,0), -1)
    # cv2.putText(frame, str(num_total2), (x3+w3-20,90),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2)

    # ////////////////////////////////////////////////////////////
    #                     ABRIR VENTANA DE VIDEO
    # ////////////////////////////////////////////////////////////

    cv2.imshow('video', frame)
    cv2.waitKey(45)
```

Código utilizado para el desarrollo de la aplicación móvil (Android Studio)

```
public class Activity_MenuPrincipal extends AppCompatActivity {

    LinearLayout linearLayoutParqueaderos;
    LinearLayout linearLayoutBibliotecas;
    LinearLayout linearLayoutPatiosComida;
    Intent intent;

    DatabaseReference mDatabase;
    ArrayList<The_Slide_Items_Model_Class> listItems;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu_principal);
```

```
        linearLayoutParqueaderos =
findViewById(R.id.LinearLayout_Parqueaderos);
        linearLayoutBibliotecas =
findViewById(R.id.LinearLayout_Bibliotecas);
        linearLayoutPatiosComida =
findViewById(R.id.LinearLayout_PatiosComida);

        mDatabase = FirebaseDatabase.getInstance().getReference();

        cargarUrlImages(mDatabase);
        setOnClickListeners();

        Bundle bundle = new Bundle();
        bundle.putParcelableArrayList("lista_url", listItems);
        intent = new Intent(this, Activity_MenuZonasMonitoreo.class);
        intent.putExtras(bundle);

    }

    public void setOnClickListeners() {
        linearLayoutParqueaderos.setOnClickListener(view -> {
            intent.putExtra("Boton_Seleccionado", "Parqueaderos");
            startActivity(intent);
        });

        linearLayoutBibliotecas.setOnClickListener(view -> {
            intent.putExtra("Boton_Seleccionado", "Bibliotecas");
            startActivity(intent);
        });

        linearLayoutPatiosComida.setOnClickListener(view -> {
            intent.putExtra("Boton_Seleccionado", "Patios de comida");
            startActivity(intent);
        });
    }

    public void cargarUrlImages(DatabaseReference db) {
        listItems = new ArrayList<>();
        db.child("Ads/").get().addOnCompleteListener(new
OnCompleteListener<DataSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<DataSnapshot> task) {
                if (!task.isSuccessful()) {
                    Log.e("firebase", "Error getting data",
task.getException());
                }
                else {
                    Log.d("firebase",
String.valueOf(task.getResult().getValue()));
                    HashMap map = (HashMap) task.getResult().getValue();
                    //System.out.println("KeySet: "+map.keySet());
                    for (Object m: map.values()) {
                        HashMap hash_map = (HashMap) m;
                        String url_imagen =
hash_map.get("url_imagen").toString();
                        String url_destino =
hash_map.get("url_destino").toString();
                        System.out.println("Url destino:
"+hash_map.get("url_destino"));
                        listItems.add(new
```

```
The_Slide_Items_Model_Class(url_imagen, url_destino));
                    }
                }
            }
        });
    }
}

public class Activity_MenuZonasMonitoreo extends AppCompatActivity {

    RecyclerView recyclerView;
    MenuCardViewZonasAdapter adapter;
    TextView txt_zonasLabel;
    DatabaseReference mDatabase =
FirebaseDatabase.getInstance().getReference();
    Context context;

    ArrayList<The_Slide_Items_Model_Class> listItems;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu_cardview);
        context = this;

        inicializarElementos();

        Bundle bundle = getIntent().getExtras();
        listItems = bundle.getParcelableArrayList("lista_url");

    }

    public void inicializarElementos() {
        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));

        String txt_boton_presionado =
getIntent().getStringExtra("Boton_Seleccionado");
        txt_zonasLabel = findViewById(R.id.zonas_label);
        txt_zonasLabel.setText(txt_boton_presionado);

        List<List> listaValues = new ArrayList<>();


mDatabase.child("/App/"+txt_boton_presionado).get().addOnCompleteListener(new
OnCompleteListener<DataSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<DataSnapshot> task) {
                if (!task.isSuccessful()) {
                    Log.e("firebase", "Error getting data",
task.getException());
                }
                else {
                    Log.d("firebase", String.valueOf(task.getResult()));
                    for (DataSnapshot postSnapshot:
task.getResult().getChildren()) {
                        ArrayList<String> lista_aux = new ArrayList<>();
                        HashMap<String, String> map;
                        lista_aux.add(postSnapshot.getKey());
                        map = (HashMap<String, String>)
postSnapshot.getValue();
```

```
                        for (String key: map.keySet()) {
                            lista_aux.add(key+"="+map.get(key));
                        }
                        listaValues.add(lista_aux);
                    }
                    //System.out.println("Lista Values: "+listaValues);
                    adapter = new MenuCardViewZonasAdapter(listaValues,
context , txt_boton_presionado, listItems);
                    recyclerView.setAdapter(adapter);
                }
            }
        });

    }
}


public class Activity_MonitoreoAforo extends AppCompatActivity {

    private ProgressBar progressBar;
    private TextView textViewProgress;
    LinearLayout linearLayoutContador;
    LinearLayout linearLayoutDisponibles;
    ImageView img_check, img_icono;
    String item_seleccionado;

    private ArrayList<The_Slide_Items_Model_Class> listItems;
    private ViewPager page;
    private TabLayout tabLayout;
    Context context;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_monitoreo_aforo);

        page = findViewById(R.id.my_pager) ;
        tabLayout = findViewById(R.id.my_tablayout);

        DatabaseReference mDatabase =
FirebaseDatabase.getInstance().getReference();

        Bundle bundle = getIntent().getExtras();
        listItems = bundle.getParcelableArrayList("lista_url");

        setImagesAds(listItems);

        linearLayoutContador = findViewById(R.id.LinearLayoutContador);
        linearLayoutDisponibles = findViewById(R.id.LinearLayoutDisponibles);
        progressBar = findViewById(R.id.progress_bar);
        textViewProgress = findViewById(R.id.text_view_progress);
        img_check = findViewById(R.id.imageView_check);
        img_icono = findViewById(R.id.imageView_icono);

        progressBar.setProgress(0);
        textViewProgress.setText("0%");

        TextView txt_aforo_actual = findViewById(R.id.txt_aforo_actual);
        TextView txt_aforo_disponible =
findViewById(R.id.txt_aforo_disponible);
        TextView txt_aforo_maximo = findViewById(R.id.txt_aforo_maximo);
```

```
        item_seleccionado = getIntent().getStringExtra("item_seleccionado");
        String item_Label = getIntent().getStringExtra("item_seleccionado");

        TextView txtView_monitoreoAforo =
findViewById(R.id.txt_MonitoreoAforoLabel);
        txtView_monitoreoAforo.setText(item_Label);
        Toast.makeText(this, item_Label, Toast.LENGTH_SHORT).show();

        ValueEventListener postListener = new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                // Get Post object and use the values to update the UI
                // Post post = dataSnapshot.getValue(Post.class);
                String aforo_actual;
                String aforo_disponible;
                String aforo_maximo;

                String[] lista_item_seleccionado = item_seleccionado.split("
");
                if (item_seleccionado.contains("Parqueadero") ||
item_seleccionado.contains("Biblioteca")) {
                    //lista_item_seleccionado[0] = "Parqueadero"
                    //lista_item_seleccionado[1] = "FIEC"
                    actualizarIcono(lista_item_seleccionado[0]);
                    aforo_actual =
String.valueOf(dataSnapshot.child("App/"+lista_item_seleccionado[0] + "s/" +
lista_item_seleccionado[1] + "/aforo_ocupados").getValue());
                    //System.out.println("Snapshot:
"+dataSnapshot.child("parqueadero_FIEC/aforo_actual").getValue());
                    txt_aforo_actual.setText(aforo_actual);
                    aforo_disponible =
String.valueOf(dataSnapshot.child("App/"+lista_item_seleccionado[0] + "s/" +
lista_item_seleccionado[1] + "/aforo_disponibles").getValue());
                    txt_aforo_disponible.setText(aforo_disponible);
                    aforo_maximo =
String.valueOf(dataSnapshot.child("App/"+lista_item_seleccionado[0] + "s/" +
lista_item_seleccionado[1] + "/aforo_maximo").getValue());
                    txt_aforo_maximo.setText(aforo_maximo);
                } else {
                    actualizarIcono(lista_item_seleccionado[0]);
                    aforo_actual =
String.valueOf(dataSnapshot.child("App/Patios de comida/" + item_seleccionado
+ "/aforo_ocupados").getValue());
                    //System.out.println("Snapshot:
"+dataSnapshot.child("parqueadero_FIEC/aforo_actual").getValue());
                    txt_aforo_actual.setText(aforo_actual);
                    aforo_disponible =
String.valueOf(dataSnapshot.child("App/Patios de comida/" + item_seleccionado
+ "/aforo_disponibles").getValue());
                    txt_aforo_disponible.setText(aforo_disponible);
                    aforo_maximo =
String.valueOf(dataSnapshot.child("App/Patios de comida/" + item_seleccionado
+ "/aforo_maximo").getValue());
                    txt_aforo_maximo.setText(aforo_maximo);
                }

updateLinearLayoutContador(Integer.parseInt(aforo_actual)*100/Integer.parseIn
t(aforo_maximo));

updateProgressBar(Integer.parseInt(aforo_actual)*100/Integer.parseInt(aforo_m
```

```
aximo));
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                // Getting Post failed, log a message
                Log.w(TAG, "loadPost:onCancelled",
databaseError.toException());
            }
        };
        mDatabase.addValueEventListener(postListener);

    }

    private void updateProgressBar(Integer progress) {
        progressBar.setProgress(progress);
        textViewProgress.setText(MessageFormat.format("{0}%", progress));
    }

    private void updateLinearLayoutContador(Integer aforo_actual) {
        if (aforo_actual >= 0 && aforo_actual < 45 ) {

linearLayoutContador.setBackgroundColor(Color.parseColor("#64D168"));
            img_check.setImageResource(R.drawable.icon_check);
        } else if (aforo_actual >= 45 && aforo_actual < 85) {

linearLayoutContador.setBackgroundColor(Color.parseColor("#FFEFD700"));
            img_check.setImageResource(R.drawable.icon_advertencia);
        } else if (aforo_actual >= 85 && aforo_actual <= 100) {

linearLayoutContador.setBackgroundColor(Color.parseColor("#FFFA695E"));
            img_check.setImageResource(R.drawable.icon_x);
        }
    }

    private void actualizarIcono(String seleccionado) {
        if (seleccionado.equals("Parqueadero")) {
            img_icono.setImageResource(R.drawable.icon_coche_01);
        } else if (seleccionado.equals("Biblioteca")) {
            img_icono.setImageResource(R.drawable.icon_libro);
        } else {
            img_icono.setImageResource(R.drawable.icon_cubiertos);
        }
    }

    private void setImagesAds(ArrayList<The_Slide_Items_Model_Class>
listItems) {
        The_Slide_items_Pager_Adapter itemsPager_adapter = new
The_Slide_items_Pager_Adapter(getBaseContext(), listItems);
        page.setAdapter(itemsPager_adapter);
        tabLayout.setupWithViewPager(page,true);

        // The_slide_timer
        java.util.Timer timer = new java.util.Timer();
        timer.scheduleAtFixedRate(new The_slide_timer(),2000,15000);
        tabLayout.setupWithViewPager(page,true);
    }

    public class The_slide_timer extends TimerTask {
        @Override
        public void run() {
```

```
                Activity_MonitoreoAforo.this.runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        if (page.getCurrentItem()< listItems.size()-1) {
                            page.setCurrentItem(page.getCurrentItem()+1);
                        }
                        else
                            page.setCurrentItem(0);
                    }
                });
            }
        }

    }
```