DESARROLLO DE UN MODELO DE GESTIÓN DE RUTAS ALIMENTADORAS PARA UN SISTEMA DE TRANSPORTE MASIVO BRT

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

PRESENTADO POR:

BRAYAN JANPAER BUITRAGO LATORRE

JOHN ALEXANDER GUTIERREZ LIZARAZO

DIRIGIDO POR:

PH.D. ROBERTO FERRO ESCOBAR

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD DE INGENIERÍA

INGENIERÍA ELECTRÓNICA

BOGOTÁ D.C.

2016

DESARROLLO DE UN MODELO DE GESTIÓN DE RUTAS ALIMENTADORAS PARA UN SISTEMA DE TRANSPORTE MASIVO BRT

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

PRESENTADO POR:

BRAYAN JANPAER BUITRAGO LATORRE

COD. 20091005086

JOHN ALEXANDER GUTIERREZ LIZARAZO

COD. 20091005085

TRABAJO DE GRADO COMO REQUISITO PARA OPTAR AL TÍTULO DE

INGENIERO ELECTRÓNICO

DIRIGIDO POR:

PH.D. ROBERTO FERRO ESCOBAR

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD DE INGENIERÍA

INGENIERÍA ELECTRÓNICA

BOGOTÁ D.C.

2016

Nota de aceptación

PhD. Roberto Ferro Escobar

Director de tesis

MSc. Roberto A. Pava Díaz

Revisor de tesis

Bogotá D.C. Colombia, 25 de noviembre de 2016

DEDICATORIA

Yo John Alexander Gutierrez, quiero dedicar este trabajo a mis padres, especialmente a mi madre y hermana que fueron mi mayor soporte para alcanzar este logro.

A Dios, quien me dio fuerzas para perseguir cada objetivo alcanzado durante mi carrera.

A mis familiares y amigos que estuvieron acompañándome en todo momento.

A mis compañeros de universidad, quienes hicieron parte de mi aprendizaje y crecimiento personal

Yo, Brayan Buitrago Latorre, dedico la culminación de este proyecto de grado a mis padres, José Buitrago y Alexandra Latorre, y mis hermanas, Alexandra Buitrago y Catalina Buitrago, por su paciencia, su compresión y apoyo constante a lo largo de este proceso educativo que culmina con este proyecto.

A mis amigos de universidad por acompañarme a través de esta gran experiencia de crecimiento personal y profesional.

AGRADECIMIENTOS

Los autores quieren expresar su agradecimiento:

Al profesor Roberto Ferro Escobar por su apoyo en el desarrollo de este proyecto.

Al Laboratorio de Investigación y Desarrollo en Electrónica y Redes, LIDER, que nos colaboró con los equipos y materiales requeridos para llevar a cabo esta tesis.

A la empresa Transmilenio S.A., quien nos suministró la información de la operación y nos posibilitó a conocer desde adentro el funcionamiento del sistema, lo que nos permitió avanzar en la ejecución del proyecto en el tiempo planeado.

A la compañía PTV AG, que nos colaboró con el software de simulación.

Índice

Índice de figuras

Índice de tablas

Introducción

Bogotá D.C, es la capital de Colombia y una de las ciudades latinoamericanas con mayor crecimiento demográfico en los últimos años. Debido a esta situación, presenta problemas relacionados con la distribución de la población, cobertura de servicios públicos, inseguridad, desempleo, entre otros.

Una de las preocupaciones que más influye en la calidad de vida de los ciudadanos es la congestión vehicular y la deficiente movilidad de la población por los principales corredores y avenidas de la ciudad. La movilidad o la falta de ella es un factor significativo para que un ciudadano tenga posibilidades de acceder a servicios básicos que le permitan alcanzar altos niveles de desarrollo humano (Roberto & Zamora-colín, 2013). Por lo tanto, la movilidad brinda una oportunidad de desarrollo económico en la ciudad.

La deficiente gestión sobre la movilidad trae consigo serios problemas, tanto para los usuarios como para los operadores de los sistemas de transporte. En el caso de los usuarios, se genera desperdicio de tiempo en trancones, transbordos y esperando rutas de buses mal programadas que no tienen en cuenta sus necesidades de movilidad, enfermedades asociadas al estrés y cardiorrespiratorias debido a la polución del aire, costos económicos relacionados a la baja productividad, etc. Por otro lado, los operadores de transporte, también ven afectada su operación, incrementado los gastos relacionados con el mantenimiento de la flota de buses, los costos salariales de los trabajadores y la infraestructura física.

Los sistemas de transporte público masivo son un pilar importante en la movilidad, debido a su alto flujo de usuarios y demanda dinámica. La gestión del servicio de estos sistemas de transporte aborda cuatro sub-problemas: planificación estratégica (rutas de tránsito y diseño de la red), planificación táctica (frecuencia de las rutas y tablas de horarios), planificación operacional

(programación de buses y de los conductores) y programación dinámica (control en tiempo real) (Luo, Zhao, Sun, Ma, & Tang, 2014). Es de especial interés la programación dinámica ya que permite administrar los recursos disponibles a las necesidades inmediatas del sistema.

En la ciudad de Bogotá se ha implementado un modelo de transporte público intermodal, soportado en un sistema BRT (Bus Rapid Transit, BRT). BRT es un modo de transporte rápido, flexible, de alto desempeño que combina elementos como carriles exclusivos, estaciones específicamente diseñadas, sistemas de operación y servicio al cliente y Sistemas de Transporte Inteligente (ITS), para ofrecer un servicio confiable, rápido, confortable y de bajo costo (Wright, 2007).

Los Sistemas Inteligentes de Transporte (Intelligent Systems Transport, ITS), permiten mejorar la gestión del servicio prestado por los BRT mediante la optimización del uso de los recursos y la aplicación de las TIC (Tecnologías de la Información y las Comunicaciones).

Planteamiento del problema

Las grandes ciudades presentan un crecimiento poblacional vertiginoso, por lo que requieren de un sistema de transporte público eficiente y con una adecuada planeación, que supla las necesidades generales de transporte de la población, especialmente en los casos donde requieren de un medio práctico para poder acceder a sus lugares de trabajo, estudio u otras actividades a realizar, y éstos van más allá de un recorrido a pie o en bicicleta. Una alternativa para cumplir con este desafío es el sistema BRT, basado en la asignación de líneas dedicadas o preferenciales para buses de rápida frecuencia de operación.

Los sistemas BRT deben ser planeados y construidos como parte de un marco general urbano buscando su integración con otros modos y opciones de transporte. Esto puede llevar a que, al igual que otros servicios de transporte como los buses tradicionales, los sistemas BRT también puedan presentar deficiencias en su operación, especialmente sobre aquellos componentes del sistema que interactúan con otros actores de la vía. Este es el caso de las rutas alimentadoras, que extienden la cobertura del sistema y optimizan la base potencial de usuarios.

Un caso particular de rutas alimentadoras como parte de un sistema BRT es el del sistema de transporte masivo Transmilenio, el medio de transporte público que más pasajeros moviliza en la ciudad de Bogotá (Indicadores & Mercado, 2015). Actualmente su modelo de servicio no monitoriza la demanda de pasajeros en tiempo real de las rutas alimentadoras, lo que generalmente provoca una alta congestión de pasajeros en horas pico, quienes deben esperar grandes períodos de tiempo en las paradas de las rutas alimentadoras ubicadas a lo largo del sistema.

Aunque la percepción ciudadana sobre la calidad del servicio del sistema Transmilenio mostró una leve recuperación durante el 2015, los buses alimentadores obtuvieron la peor calificación,

empeorando incluso si se compara con respecto al año inmediatamente anterior. Cerca del 45% de los usuarios de Transmilenio utilizan las rutas alimentadoras, lo que evidencia la importancia de garantizar una buena prestación de servicio en este componente, ofreciendo mayor comodidad y optimizando los tiempos de recorrido de las rutas (de Periódico El Tiempo).

Se evidencia la necesidad de implementar un nuevo modelo que responda de manera más rápida a las necesidades del sistema, que tenga en cuenta la demanda eventual de los usuarios y los recursos disponibles por parte del operador.

¿Es posible mediante un modelo de gestión de rutas alimentadoras, mejorar la respuesta a la demanda de pasajeros durante las horas pico de un sistema de transporte masivo BRT?

Justificación

En este proyecto se diseñará un prototipo de un módulo de adquisición de datos que permita obtener información de la demanda de usuarios de un sistema de transporte público BRT, tomando como caso de estudio la demanda de usuarios del servicio de buses alimentadores del sistema de transporte Transmilenio para la estación central Banderas, con el fin de poder analizar los datos obtenidos y con esto aumentar la eficiencia en la prestación del servicio, por parte del operador de transporte.

El sistema de transporte es un eje vital para el desarrollo de una ciudad, razón por la cual, la calidad del servicio debe ser orientada a buscar altos niveles, y la tecnología actual tiene el potencial necesario para contribuir a que esto sea así.

La propuesta de Sistemas de Transporte Inteligente esta soportada en aplicaciones que permitan que los usuarios estén informados del estado y la dinámica del sistema de transporte; en coherencia con esto, los usuarios del sistema de transporte masivo suministrarán de manera transparente al operador, información asociada a su ruta de destino para así poder mejorar la gestión del servicio, ya que el operador de transporte podrá responder de manera más ágil y eficiente a la demanda real de pasajeros, y podrá disponer mejor de los recursos involucrados, basado en los antecedentes de la demanda.

Este modelo de gestión permitirá mejorar la planeación del servicio, en el largo, mediano y corto plazo. Desde el punto de vista estratégico, será posible analizar el comportamiento de los usuarios del sistema para hacer una previsión de la demanda, reevaluar los alcances, medir el desempeño del servicio por medio de factores claves y proyectar mejoras continuas del sistema.

También se podrán evaluar las fortalezas y debilidades de la gestión del servicio de buses alimentadores, y así identificar las acciones a realizar dependiendo de las condiciones del sistema

y la demanda; de esta manera se podrán desarrollar mecanismos de acción para aquellas situaciones donde se identifique una anomalía en el comportamiento de la demanda. Los estudios a nivel estratégico están relacionados con la planificación a largo plazo de la red, análisis de comportamiento de los clientes y la previsión de la demanda (Pelletier, 2009).

Según la Formulación Del Plan Maestro De Movilidad Para Bogotá D.C, el sistema de transporte público debe ser capaz de ajustarse a la dinámica urbana propia, buscando la organización de la gestión del servicio para aumentar el rendimiento y reducir los costos del servicio. También se reconoce la necesidad de realizar modificaciones y adaptaciones en la estructura operacional del servicio para mejorar la eficiencia en la prestación del servicio (Secretaría Transito y Transporte, 2006).

Con respecto a la inclusión de la tecnología y el continuo mejoramiento de ésta en la estructura operacional del servicio de transporte, se infiere que "desde el punto de vista de la demanda, y del mismo modo que en otras áreas vinculadas a las Smart Cities, las soluciones de transporte público evolucionan adaptándose a las necesidades y expectativas de los clientes, y no al revés, mediante la aparición de soluciones como planificadores de viaje, apps de información del servicio en tiempo real, esquemas de uso compartido, o la exigencia de servicios bajo demanda. El tipo de cliente de los servicios de transporte evoluciona, con un perfil cada vez más flexible y multimodal." (Pérez, Velásquez, Fernandez, & Dorao, 2012).

Los sistemas de transporte inteligente usan tecnologías que permitan la interconexión y gestión de la información, mediante recolección y análisis de datos en tiempo real; estas tecnologías de Big Data hacen factible el procesamiento de los datos correspondientes a los patrones de las variables del sistema en un tiempo corto para que se puedan tomar decisiones inmediatamente sobre el dinamismo de la demanda.

Con la adquisición de datos se puede corroborar la eficiencia de los sistemas de predicción de la demanda que sean implementados en el sistema de transporte, y se puede efectuar una calibración rápida de la matriz origen-destino**.** (Kostic & Gentile, 2015)**.**

El desarrollo de este proyecto propondrá un método de adquisición de los datos de los usuarios, que permitirá monitorizar la dinámica del sistema de transporte, más específicamente en el servicio de buses alimentadores, apuntando a que el tiempo de espera de los pasajeros dentro del sistema sea el menor posible.

La aplicación de este módulo deberá hacer uso de una base de datos en la cual se almacenen y relacionen los distintos factores que se presentan en un sistema de transporte masivo, como el número de personas dentro del sistema en las diferentes horas del día, los tiempos de espera, así como la variación de los patrones de demanda en función de los diferentes días de la semana. En potencia, aplicando técnicas de Big Data, estos datos podrían ser relacionados con el clima para predecir futuros comportamientos de la demanda. El hecho que existan datos de la demanda de pasajeros del sistema de transporte masivo, en donde se relacionen el número de pasajeros, fechas y horas, permite además, que se puedan simular posibles soluciones para aumentar la eficiencia de la gestión del servicio (Jin & Wang, 2008)**.** Los datos que se recolecten con un módulo de estos pueden permitir impulsar un campo de investigación orientado al análisis de la demanda dinámica.

En potencia esta propuesta beneficiaria al 45% de los usuarios del sistema, que corresponde a la proporción de los usuarios que usan el servicio de rutas alimentadoras, pues se busca que la calidad de vida aumente para los pasajeros y sus familias, ya que esto tendría connotaciones en la salud de los usuarios y la forma en la que invierten su tiempo.

Objetivos

Objetivo general

Desarrollar un modelo de gestión de rutas alimentadoras para un sistema de transporte masivo BRT.

Objetivos específicos

- Efectuar un estudio comparativo de las etiquetas disponibles para la elaboración del proyecto, destacando las ventajas y características de cada una.
- Formular indicadores clave de desempeño (Key Performance Indicators, KPI) que permitan evaluar la gestión del servicio dentro del sistema Transmilenio.
- Determinar la demanda de pasajeros que utilizan un servicio de transporte masivo BRT en las paradas de buses alimentadores, para definir los parámetros de operación del sistema de gestión.
- Desarrollar una aplicación web para gestionar las rutas alimentadoras de un sistema BRT a corto y mediano plazo, que responda a la demanda de pasajeros.
- Diseñar un sistema de gestión de tráfico robusto y adaptable a los cambios del flujo de información, soportado en ITS.

.

Antecedentes

La integración de la tecnología NFC a diversos campos de aplicación ya ha sido investigada y analizada por varios años. En el sector del transporte sobresale en aplicaciones como registros meteorológicos, mapas de ubicación, ofertas de productos con descuentos, tiempos de llegada de rutas de buses, servicios de taxis, llamadas de emergencia, transferencia de llamadas, entre otros (Gautam, Kumar, & Gupta, 2014).

Dentro de los proyectos pilotos más importantes encontramos: el proyecto de servicio de tiquetes NFC llevado a cabo por Telecom Italia y ATM para permitir a sus clientes utilizar teléfonos móviles compatibles con NFC para acceder a la red de transporte público y efectuar la compra y validación de tiquetes (Clark, 2009). En (Finzgar & Trebar, 2011), describen la implementación de un sistema que permite el uso de los celulares de los pasajeros para adquirir tiquetes electrónicos.

En el campo de la gestión y administración de tráfico vehicular se han enfocado en el uso de la tecnología RFID, sobre la cual se basa NFC pero que posee importantes diferencias con respecto al alcance y la transmisión de datos. Se realizó una investigación basada en la programación dinámica de los buses basada en el seguimiento de pasajeros mediante agentes inteligentes dentro de un sistema de transporte público (Hamilton & Sankaranarayanan, 2013). Además, se ha propuesto usar la tecnología RFID embebida en las tarjetas inteligentes de pago para almacenar datos de flujo de pasajeros, proveyendo información para la administración de las operaciones de tránsito y planificación en tiempo real (Ferreira, Gouveia, Facchini, Pokorny, & Dias, 2012).

Se ha propuesto implementar una plataforma para sistemas de transporte público que utilice las etiquetas y lectores de tecnología RFID combinados con señales de tráfico variables (Variable

Message Signs, VMS) con el fin de ofrecer información en tiempo real a los operadores de transporte (Assaf & Williams, 2011).

En la ciudad de Sao Paulo se sugirió la posibilidad de embeber etiquetas RFID dentro de las tarjetas inteligentes "Bilhete Único" utilizadas como medio de pago en el transporte público desde 2004, con el fin de identificar y almacenar la información de las rutas de cada pasajero y su ubicación actual. Por otro lado, las paradas de buses también tendrían etiquetas RFID para determinar las rutas y tiempos de recorrido de cada bus (Fereira et al., 2009).

Una herramienta utilizada en la gestión y planificación de transporte es la matriz Origen - Destino. Se ha propuesto crear la matriz, partiendo de los datos entregados por los usuarios a través de dispositivos con tecnología Bluetooth, analizando una ruta del sistema Transmilenio de la ciudad de Bogotá (Cañon-Lozano, Melo-Castillo, Gomez-Perilla, Banse, & Herrera-Quintero, 2013). También, se ha realizado un análisis con los Sistemas Automáticos de Colección de Datos (Automated Data CollectionSystems, ADCS) y la Localización Automática de Vehículos (Automatic Vehicle Location, AVL) para la estimación de la demanda de pasajeros de una ruta de bus, registrando la permanencia de éstos en el vehículo a lo largo del recorrido, lo que permite a los gestores del sistema de transporte alcanzar altos niveles de eficiencia a largo plazo (Wang, Attanucci, & Wilson, 2011).

Por otro lado, se planteó tener en cuenta el estado actual de tráfico mediante el uso de sensores de punto, sensores punto-punto y sensores de área, de gran uso para aumentar la precisión de la asignación estática de tráfico (Kostic & Gentile, 2015). En la ciudad de Porto, se diseñó una metodología para mejorar los resultados de la estimación de la matriz OD por medio de un algoritmo que tiene en cuenta los datos asociados a la ruta, el vehículo, la tarjeta de transporte utilizada, la hora y el lugar donde inicia el viaje, con resultados aceptables en la estimación,

encontrando mayor precisión en la inferencia de los destinos de los usuarios, y a su vez disminuyendo la tasa general (Nunes, Galvao Dias, & Falcao e Cunha, 2015).

Un sistema inteligente enfocado en dar respuesta a la demanda de usuarios busca satisfacer las necesidades de transporte ajustando dinámicamente la ruta de los taxis soportado en el modelo del Sistema de Transporte Receptivo a la Demanda (DRT) (Jin & Wang, 2008). El DRT se ha propuesto como un medio de transporte masivo a gran escala que ofrece un servicio rentable para el operador y beneficioso para el usuario por su capacidad de tolerar cambios inesperados en la demanda que afectan el rendimiento del sistema, y por la posibilidad de ofrecer una promesa de calidad de servicio (Jokinen, Sihvola, Hyytiä, & Sulonen, 2011).

Marco teórico

¿Qué es la tecnología NFC?

NFC es una tecnología de corto rango, alta frecuencia, bajo ancho de banda y comunicación inalámbrica entre dos dispositivos compatibles, que ha emergido a partir de la convergencia de tecnologías de identificación sin contacto como RFID y tecnologías de comunicaciones móviles como Bluetooth y Wifi (Ilyas, 2012).

Su importancia deriva de la gran variedad de aplicaciones potenciales donde se puede implementar como el pago electrónico, identificación, control de acceso, distribución de contenido, transferencia de datos, transporte, domótica, entre otras. De esta manera, y debido al amplio rango de áreas donde puede utilizarse, ofrece una atractiva oportunidad para muchas instituciones académicas, organizaciones y compañías.

Motivación tecnológica

La computación ubicua es comprendida como la interacción más natural que puede existir entre humanos y máquinas, permitiendo que éstas se integren a nuestra vida diaria ajustándolas a nuestras necesidades. En este campo, NFC surge como una nueva tecnología de interconexión e identificación sin contacto físico que nos permite la interacción con múltiples dispositivos a una corta distancia.

La principal motivación para el desarrollo de la tecnología NFC es la integración de información personal y privada como las transacciones financieras dentro de los dispositivos móviles. Sin embargo, y debido a la sensibilidad que conlleva el intercambio de este tipo de información, la seguridad es la mayor preocupación y el rango de comunicación inalámbrica proveído por la tecnología RFID es considerado demasiado amplio, lo que implica la utilización de mecanismos para proteger la información de accesos no autorizados a la información

contenida en las etiquetas que, aunque no posean una fuente de alimentación eléctrica, pueden ser leídas alrededor por otros dispositivos que se encuentren a unos 10m alrededor (Coskun, Ok, & Ozdenizci, 2012).

Evolución de NFC

Aunque la tecnología RFID puede considerarse la precursora de NFC, existen otras tecnologías que surgieron anteriormente y han sido igual de importantes, abonando el camino para el desarrollo de la tecnología NFC.

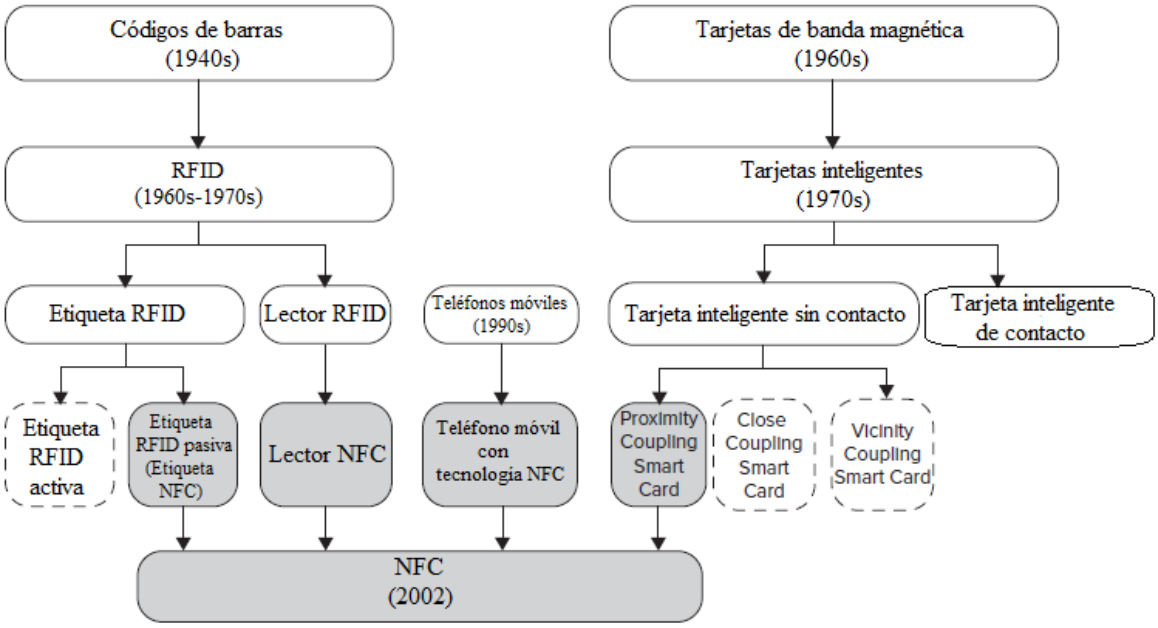

Figura 1. Evolución de la tecnología NFC (Bash, 2015)

Código de barras

El código de barras es una representación visual que contiene datos acerca del objeto sobre el cual es colocado. La información es capturada por escáneres ópticos llamados lectores de barras y transferidos a los dispositivos a los que están conectados los lectores. Inicialmente aparecieron

códigos de barras unidimensionales (1D) cuyos datos eran representados variando el ancho y espaciado de las líneas paralelas.

Más tarde, surgieron los códigos de barras bidimensionales (2D) con una mayor capacidad de almacenamiento que los códigos de barras lineales, permitiendo contener un único valor para cada ítem específico dentro de cada grupo de productos etiquetados.

Figura 2. Código de barras tipo UPC-A

RFID (Radio – Frequency Identification)

RFID es una tecnología que usa la comunicación por medio de ondas de radio para intercambiar datos entre un lector RFID y una etiqueta electrónica RFID generalmente conectada a un objeto para propósitos de seguimiento e identificación. Así pues, un sistema RFID está compuesto de dos componentes principales: el transpondedor, que es el componente que se localiza en un producto u objeto para ser identificado, y el lector que lee o escribe los datos en el transpondedor.

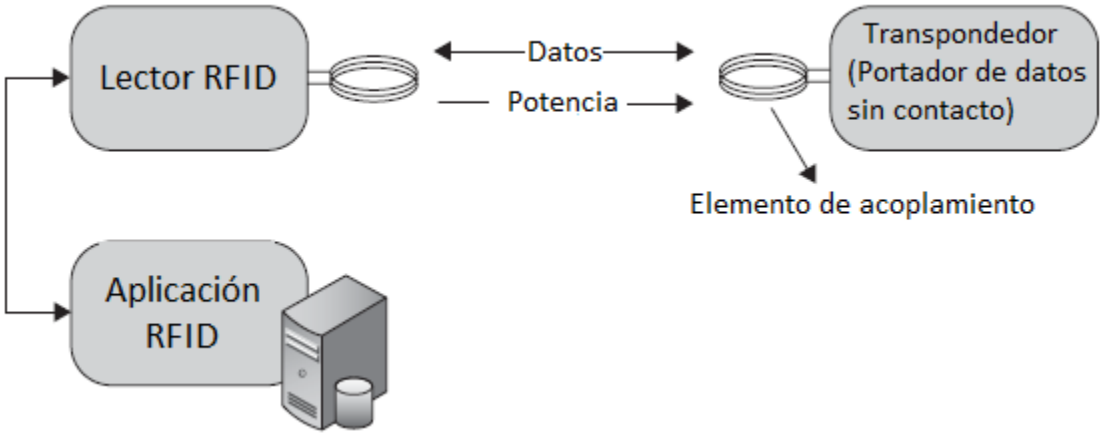

Figura 3. Sistema RFID y sus componentes (Coskun, Kerem, & Ozdenizci, 2013).

Existen dos tipos de etiquetas RFID: pasivas y activas. Las etiquetas pasivas no poseen fuente propia por lo que son energizadas mediante una señal de campo en radiofrecuencia RF, tienen embebidos un circuito integrado y una antena (Coskun et al., 2012). Pueden ser leídas en un rango que va de los 10cm a varios metros, dependiendo de la frecuencia de radio seleccionada y el diseño y tamaño de la antena.

Por otro lado, las etiquetas activas tienen su propia fuente de alimentación utilizada para energizar algunos circuitos integrados que generan una señal de uso externo, son mucho más confiables que las etiquetas pasivas por su capacidad de establecer una sesión con un lector a mayores distancias. Tienen un rango más amplio y mayor memoria que las etiquetas pasivas, aunque son más grandes y costosas.

La tecnología RFID ha sido utilizada en una amplia variedad de aplicaciones. Entre estas se encuentran: sistemas de inventario, implantes humanos, identificación animal, seguimiento mediante chips en casinos, cuartos de operación en hospitales, entre otras (Bash, 2015).

Tarjetas de banda magnética

Una tarjeta de banda magnética contiene un espacio de almacenamiento digital donde los datos son cargados durante la fase de fabricación. Generalmente, es un elemento de sólo lectura mediante contacto físico al pasar la tarjeta por un dispositivo con cabeza de lectura magnética.

Tarjetas inteligentes

Una tarjeta inteligente es un dispositivo que contiene un circuito integrado embebido con una unidad de memoria incorporada y en su mayoría posee un microcontrolador o dispositivo equivalente. No contiene fuente de alimentación por lo que es energizada mediante un dispositivo externo o a través del lector con el cual interactúa. Un sistema de tarjetas inteligentes se compone de las tarjetas, los lectores de tarjetas y un sistema back-end.

En términos del mecanismo de operación, las tarjetas inteligentes pueden clasificarse en tres grupos: tarjetas de contacto, sin contacto y modelos híbridos.

La tarjeta de contacto se comunica con el lector mediante contacto físico, de la misma manera que recibe la energía requerida. Es embebida con un micromódulo que posee un circuito integrado de silicio, el cual posee memoria y un microprocesador.

En el caso de la tarjeta sin contacto, la comunicación con el lector se lleva a cabo mediante una interfaz de radiofrecuencia cuando los dos dispositivos están suficientemente cerca. Por un lado, esto asegura una mayor seguridad en la comunicación y además, permite una mayor transferencia de energía del dispositivo activo al pasivo.

En el último grupo se encuentran los modelos híbridos, los cuales poseen dos chips independientes y que no se encuentran conectados entre sí. Uno de los chips es utilizado para una interfaz de contacto y el otro para la interfaz sin contacto.

Las tarjetas más populares son las tarjetas sin contacto basadas en la norma ISO/IEC 14443. Operan a una distancia de hasta 10cm, lo cual las convierte en la mejor elección en un amplio rango de áreas de aplicación, desde la salud hasta el entretenimiento. Las tarjetas más famosas desarrolladas bajo este estándar son MIFARE, Calypso y FeliCa.

NFC como nueva tecnología

En el año 2002, Philips y Sony presentaron conjuntamente la tecnología NFC. Fue adoptada como estándar por la Organización Internacional para la Estandarización (ISO) y la Comisión Internacional de Electrotecnia (IEC) en el año 2003.

NFC es una tecnología de comunicación inalámbrica bidireccional que funciona sobre el espectro de 13.56MHz de RFID. Permite velocidades de transferencia de 106, 212 y 424kbps.

Opera en diferentes modos: lectura – escritura, punto a punto, y en modo emulación. Cada modo de operación utiliza interfaces de comunicación específicas (ISO/IEC 14443, FeliCa, NFCIP-1) en la capa de transmisión por radiofrecuencia así como el manejo de diversas técnicas y requerimientos de operación y diseño (Coskun et al., 2012).

Arquitectura de NFC

Dentro de la arquitectura de la tecnología NFC debemos tener en cuenta varias capas. Encontramos la capa física, donde la CPU y las señales de radio realizan la comunicación. En el medio, está el empaquetamiento de datos y las capas de transporte, luego vienen las capas de formato y finalmente el código de aplicación.

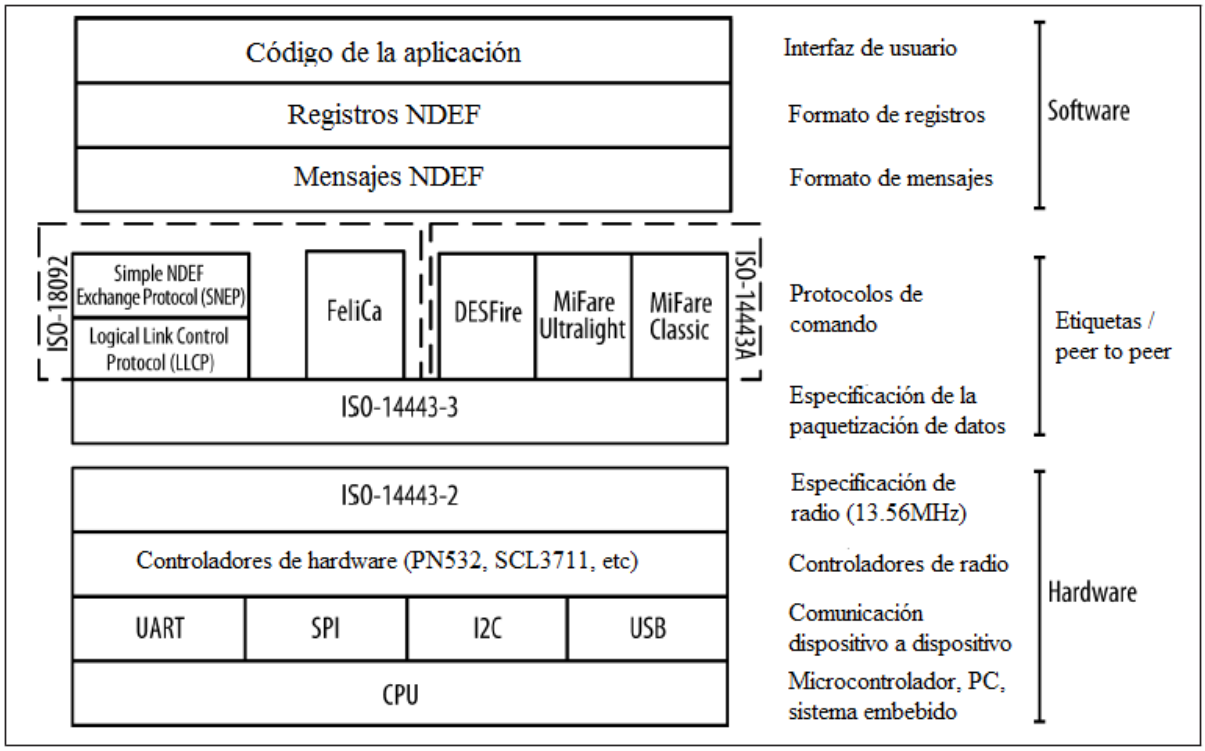

Figura 4. Pila de protocolos NFC (Bash, 2015)

Capa física

NFC trabaja sobre la especificación de radio de RFID, ISO-14443-2, que describe radios de operación de corto rango a 13.56MHz.

Luego viene la capa que describe la fragmentación de bytes de datos sobre radio, ISO-14443-3. Cualquiera de los radios que se puede usar son componentes de hardware separados, tanto dentro del teléfono móvil o tableta, o adjuntos al microcontrolador o computador personal. Estos se comunican con el procesador del dispositivo usando uno o más protocolos seriales estándar: Universal Asynchronous Receive-Transmit (UART), Serial Peripheral interface (SPI), Inter Integrated Circuit Communication (I2C), o Universal Serial bus (USB) (Bash, 2015).

La ISO (Organización Internacional de Normalización) e IEC (Comisión Electrotécnica Internacional) definieron la norma ISO/IEC 14443 para la estandarización para el uso de Tarjetas de Circuito Integrado sin Contacto, Tarjetas de Identificación y Tarjetas de Proximidad. Esta norma se compone de 4 partes:

- Parte 1: Características físicas
- Parte 2: Potencia de radiofrecuencia y la señal de interfaz
- Parte 3: Inicialización y anticolisión
- Parte 4: Protocolos de transmisión

Parte 1: Características Físicas:

Los fabricantes de las tarjetas de circuito integrado sin contacto deben garantizar que sus productos (Tarjetas) soporten la exposición a ciertos tipos de ambientes o eventos. Las tarjetas deben funcionar normalmente en exposición a los rayos ultravioleta de la luz solar. También deberán funcionar después de que sean expuestas a una radiación media de rayos X, con una energía de 100 keV (kilo-electrón Voltios), de una dosis acumulada de 0,1 Gy (Rayos X y gamma) por año; esto corresponde al doble de exposición máxima aceptada en un humano

anualmente. Deberá soportar una deflexión máxima de 10mm y 20mm en el eje corto y el eje largo respectivamente. La tarjeta funcionara normalmente después de ser expuesta un campo magnético de 12 A / m en 13,56 MHz y a los niveles de campo eléctrico y magnético referenciados en la tabla 1. El rango de operación de temperatura debe estar entre 0°C y 50°C (Standardization, 2000).

Tabla 1. Niveles de exposición que debe soportar la tarjeta sin alterar su funcionamiento (Standardization, 2000).

Rango de Frecuencia (MHz)	Campo Magnético Promedio (A/m)	Campo Eléctrico Promedio (V/m)	Tiempo Promedio (Minutos)
0,3-3,0	1,63	0,614	6
3,0-30	4,98/f *	1842/f[1]	6
30-300	0,163	61,4	6

Parte 2: potencia de radiofrecuencia y la señal de interfaz

El diálogo inicial entre el Dispositivo de Acoplamiento de Proximidad (PCD, por sus siglas en inglés, Proximity Coupling Device) y la Tarjeta de Proximidad (PICC, por sus siglas en inglés, Proximity Integrated Circuit Card) se llevará a cabo a través de las siguientes operaciones consecutivas (Standardization, 2001a):

- Activación de la PICC por el campo de operación de radiofrecuencia (RF) del PCD
- La PICC espera en silencio por un comando del PCD.
- Transmisión de un comando del PCD
- Transmisión de una respuesta por la PICC.

El PCD se producirá un campo de RF energizante que se acopla a la PICC para transferir energía, y el cual sera modulado para la comunicación. La frecuencia de operación será 13,56

[1] Corresponde a la frecuencia en MHz

MHz ± 7 kHz. El campo magnético generado por el PCD debe ser mínimo de 1,5 A/m (RMS) y no debe exceder los 7,5 A/m (RMS).

Existen dos interfaces de comunicación, Tipo A y Tipo B. El PCD se alternará entre los métodos de modulación hasta detectar la presencia de una PICC de tipo A o B. Sólo una interfaz de señales de comunicación puede ser activo durante una sesión de comunicación hasta la desactivación por el PCD o remoción del PICC (Standardization, 2001a). En la Figura 5 se observa las señales de comunicación para cada uno de las interfaces de comunicación.

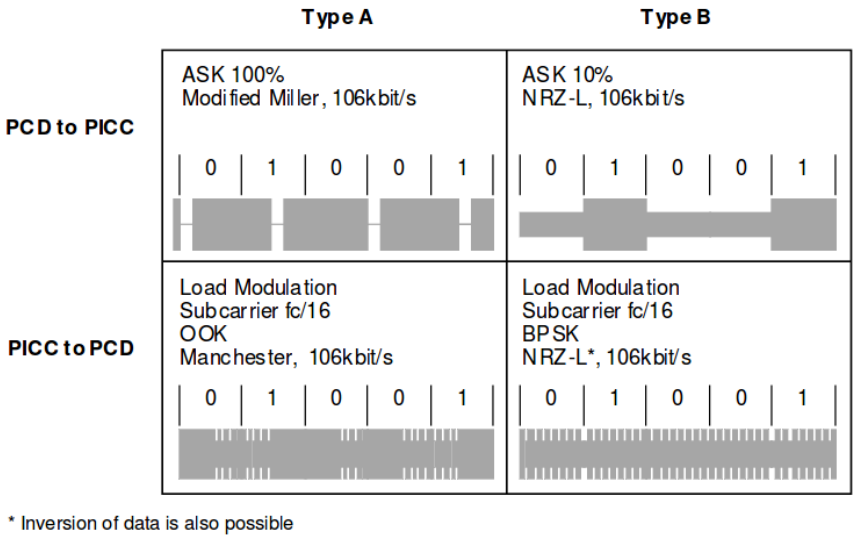

Figura 5. Ejemplo de señales de comunicación para interfaces de tipo A o Tipo B (Standardization, 2001a).

La tasa de transmisión durante la inicialización y anticolisión debe ser de 106kbit/s, que corresponde a 13,56 MHz dividida en 128. En la comunicación de PICC a PCD la sub-portadora es de 847 kHz, correspondiente a 13,56MHz/16.

Parte 3: Inicialización y anticolisión

Cuando una PICC es expuesta a un campo que no corresponde a su modulación de operación, esta será capaz de aceptar una solicitud de comunicación dentro de 5ms. Un PCD envía comandos de solicitud Tipo A y Tipo B en busca de un ATQ. Los comandos de solicitud

enviados por el PCD son REQA (por sus siglas en inglés, Request Command Type A) y REQB (por sus siglas en inglés, Request command Type B) (Standardization, 2001b).

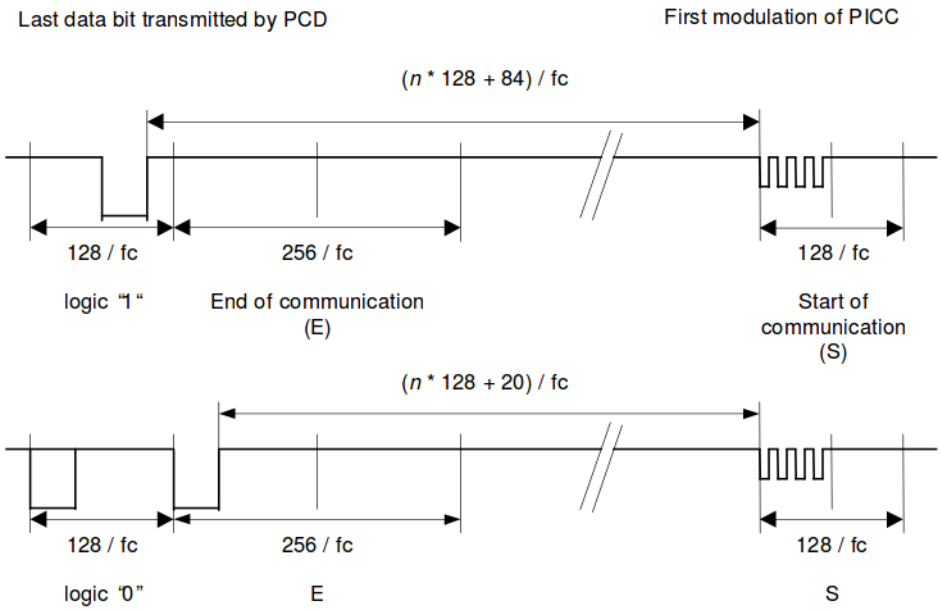

Figura 6. Tiempo de Retardo de Trama entre PCD y PICC (Standardization, 2001b).

En la Figura 6 se observa el Tiempo de Retardo de Trama (FDT, por sus siglas en inglés, Frame Delay Time) entre el último dato transmitido por un PCD y la primera modulación de una PICC. Como se puede notar el FDT depende de un entero *n* y del bit que finalice la trama. En la Tabla 2 se encuentra la forma para calcular el FDT.

Tabla 2. Cálculo de FDT.

Tipo de Comando	n (Valor entero)	FDT	
		Ultimo bit = 0	**Ultimo bit = 1**
REQA Command WAKE-UP Command ANTICOLLISION Command SELECT Command	9	1236 / fc	1172 / fc
Otro Comando	≥9	(n * 128 + 84) / fc	(n * 128 + 20) / fc

El Tiempo de Solicitud de Guarda (RGT, por sus siglas en inglés, Request Guard Time) se define como el tiempo mínimo entre los bits de inicio de dos comandos de solicitud consecutivos. Tiene el valor 7000 / fc (Standardization, 2001b).

La trama REQA y la trama WAKE-UP son usadas para iniciar la comunicación. En la Figura 8 se observa el mensaje enviado:

- Primero se envía un bit de inicio de trama S.
- Luego se envían 7 bits de la trama REQA o la trama WAKE-UP, enviando primero el bit menos significativo (LSB, por sus siglas en inglés, Least Significant Bit) y finalizando con el bit más significativo (MSB, por sus siglas en inglés, Most Significant Bit). Para un REQA el dato corresponde a un '0x26' y para un WAKE-UP Request el dato corresponde a un '0x52'.
- Finalmente, se envía un bit de finalización E.

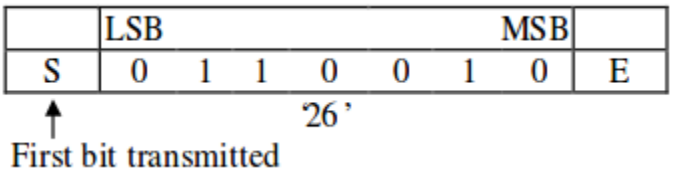

Figura 7. Trama de un REQA (Standardization, 2001b).

Las tramas estándar son usadas para intercambiar datos entre el PDC y la PICC. El envío de estas se describe a continuación:

- Primero se envía un bit de inicio de trama S.
- Se envía un bit de paridad por cada 8 bits de datos. Así que el tamaño de estos daros corresponde a n * (8 data bits + odd parity bit).
- Finalmente se envía un bit de finalización E.

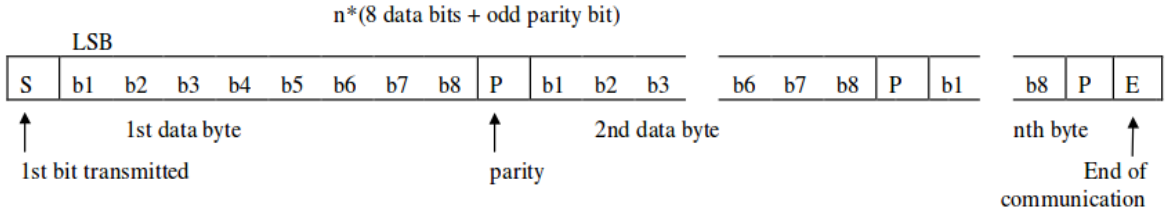

Figura 8. Trama Estándar (Standardization, 2001b).

Una colisión es detectada cuando dos o más PICC intentan transmitir a la PCD. Las tramas de bits orientadas a la anticolisión solo son usadas durante los bucles de tramas de bit anticolisión, estas son tramas estándar de 7 Bytes divididas en dos partes:
Parte 1 es transmitida desde el PCD a la PICC y la parte 2 es transmitida desde la PICC hacia el PCD.

La longitud de las dos partes debe ser de 56 bits, 7 Bytes. La longitud mínima de la parte 1 debe ser 16 bits, o sea que la máxima longitud de la parte 2 debe ser 40 bits. La longitud máxima de la primera parte debe ser 55 bits, o sea que la mínima longitud de la segunda parte debe ser

Existen dos casos de división de estas partes: FULL BYTE y SPLIT BYTE.

FULL BYTE: es cuando se divide después de un Byte completo. Se agrega un bit de paridad después del último bit y se agrega el bit de finalización, como se observa en la Figura 9.

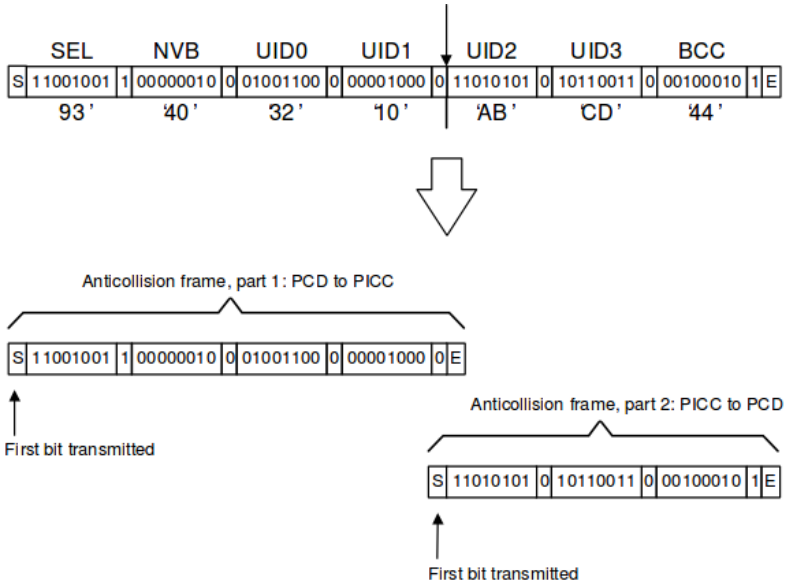

Figura 9. FULL BYTE, trama dividida después del cuarto Byte completo (Standardization, 2001b).

SPLIT BYTE: Cuando se divide la trama dentro de un byte. En este caso no se agrega un bit de paridad, solo el bit de finalización, como se observa en la Figura 10.

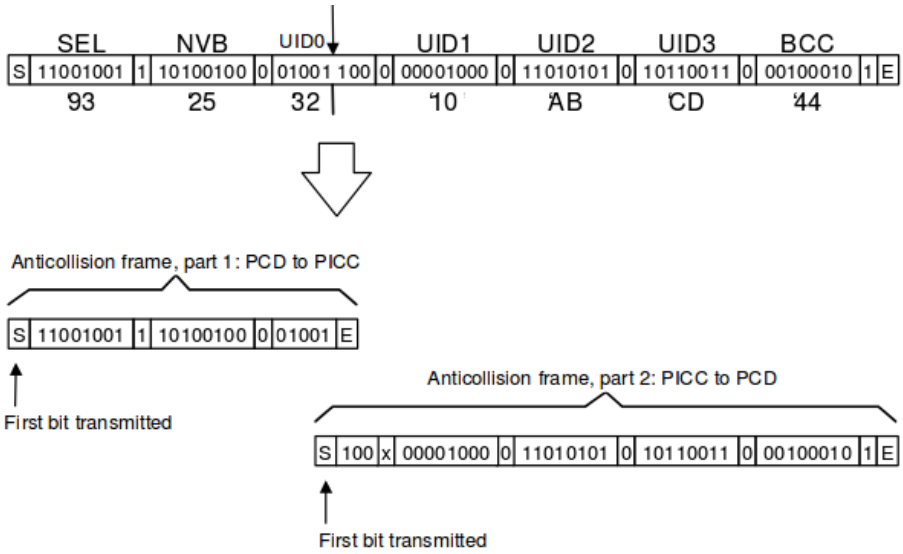

Figura 10. SPLIT BYTE, división de la trama en el tercer byte (Standardization, 2001b).

En la Figura 11 se puede observar el protocolo de colisiones para una PICC de tipo A.

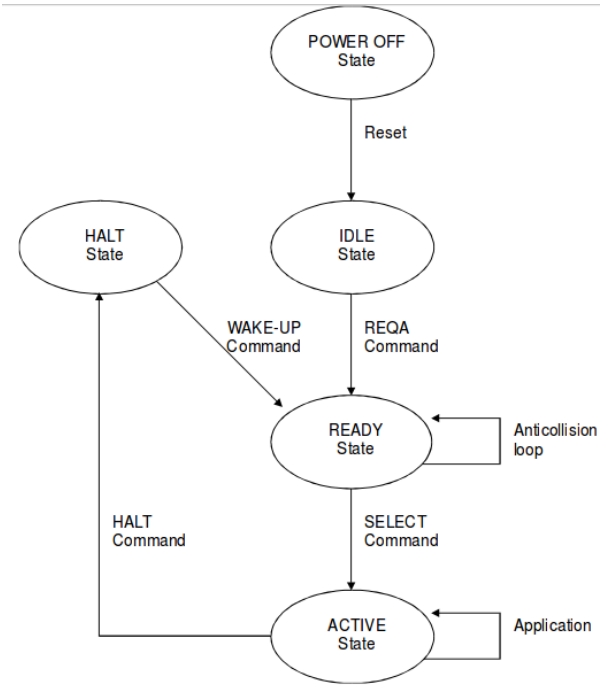

Figura 11. Estados de una PICC durante un protocolo anticolisión (Standardization, 2001b).

Estado POWER OFF: En este estado la PICC no se encuentra energizada.

Estado IDLE: Después de ser energizado por el tiempo necesario, el PICC entra en el estado IDLE y es capaz de demodular y reconocer las tramas REQA y WAKE-UP enviadas por el PCD.

Estado READY: Se entra a este estado cuando se ha validado la trama REQA y la trama WAKE-UP. Se sale de este estado cuando se sale cuando el PICC es seleccionado con su Identificador Único (UID, por sus siglas en inglés, Unique Identificator.

Estado ACTIVE: Se entra a este estado cuando se obtiene la UID completa de la PICC.

Estado HALT: Cuando se encuentra en este estado, la PICC se encuentra detenida y no interactúa hasta que reciba un comando WAKE-UP que lo envíe a un estado READY.

Para que la PICC cambie de estado se le deben enviar ciertos comandos. El PCD hace uso de los siguientes comandos para establecer comunicación con la PICC:

Comando REQA: El Comando REQA es enviado por el PCD para sondear el campo de la PICC de tipo A.

Comando WAKE-UP: Este comando lo envía el PCD para cambiar los PICC que se encuentran en el estado HALT a un estado READY.

En la Tabla 4 se encuentra los datos correspondientes a los comandos REQA y WAKE-UP.

Tabla 3. Comandos REQA y WAKE-UP.

b7	b6	b5	b4	b3	b2	b1	Comando
0	1	0	0	1	1	0	REQA = 0x26
1	0	1	0	0	1	0	WAKE-UP = 0x52

Comando SELECT, Comando ANTICOLLISION: Estos comandos son utilizados durante un bucle anticolisión. El comando consiste de las siguientes partes:

- Código Select, SEL, de 1 byte.
- Numero de bits validos (NVB, por sus siglas en inglés, Number of Valid Bits) de 1 byte.
- De 0 a 40 bits de datos que corresponden al UID, en concordancia con el NVB.

Mientras ocurre el bucle anticolisión se va conformando el UID por niveles, estos niveles son llamados Nivel de Cascada (CL, por sus siglas en inglés, Cascade Level). SEL especifica el nivel de cascada Cln.

Comando HALT: Este comando consiste en 4 bytes y debe ser enviado usando una trama estándar. La codificación CRC_A y el proceso de verificación se define en la Recomendación UIT-T V.41, párrafo 2. El polinomio generador utilizado para generar los bits de control es x^16 + x^12 + x^5 + 1. El valor inicial será '6363'. El CRC_A se añadirá a los bytes de datos y se transmite a través de las tramas estándar.

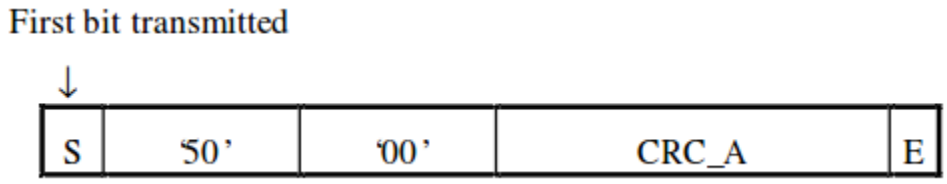

Figura 12. Comando HALT (Standardization, 2001b).

Secuencia de Selección (Select Sequence): Con esta secuencia se obtiene el UID de la PICC y se selecciona dicha PICC para mantener comunicación.

Respuesta de solicitud ATQA (Answer To Request): Después de que el PCD transmite un REQA, todos los PICC cercanos al campo responden sincrónicamente con un ATQA, y codifica el tipo de anticolisión aplicable en 2 bytes. Nota: RFU = Reservado para usos futuros (Recerve for Future Use).

Tabla 4. Código ATQA.

MSB															**LSB**
b16	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1
RFU								Tamaño de la trama UID		RFU	Trama de bits anticolisión				

Los bits b8 y b7 definen el tamaño del UID como se muestra en la Tabla 5. Si uno de los bits b5, b4, b3, b2 o b1 es 1, indica la trama anticolisión como se muestra en la Tabla 6.

Tabla 5. Código de bit 8 y bit 7 de la trama anticolisión

b8	**b7**	**Significado**
0	0	Tamaño de UID: Sencillo
0	1	Tamaño de UID: Doble
1	0	Tamaño de UID: Triple
1	1	RFU

Tabla 6. Código de bits (5-1) para la trama anticolisión.

b5	b4	b3	b2	b1	Significado
1	0	0	0	0	Trama de bits anticolisión
0	1	0	0	0	Trama de bits anticolisión
0	0	1	0	0	Trama de bits anticolisión
0	0	0	1	0	Trama de bits anticolisión
0	0	0	0	1	Trama de bits anticolisión
Otros Valores					RFU

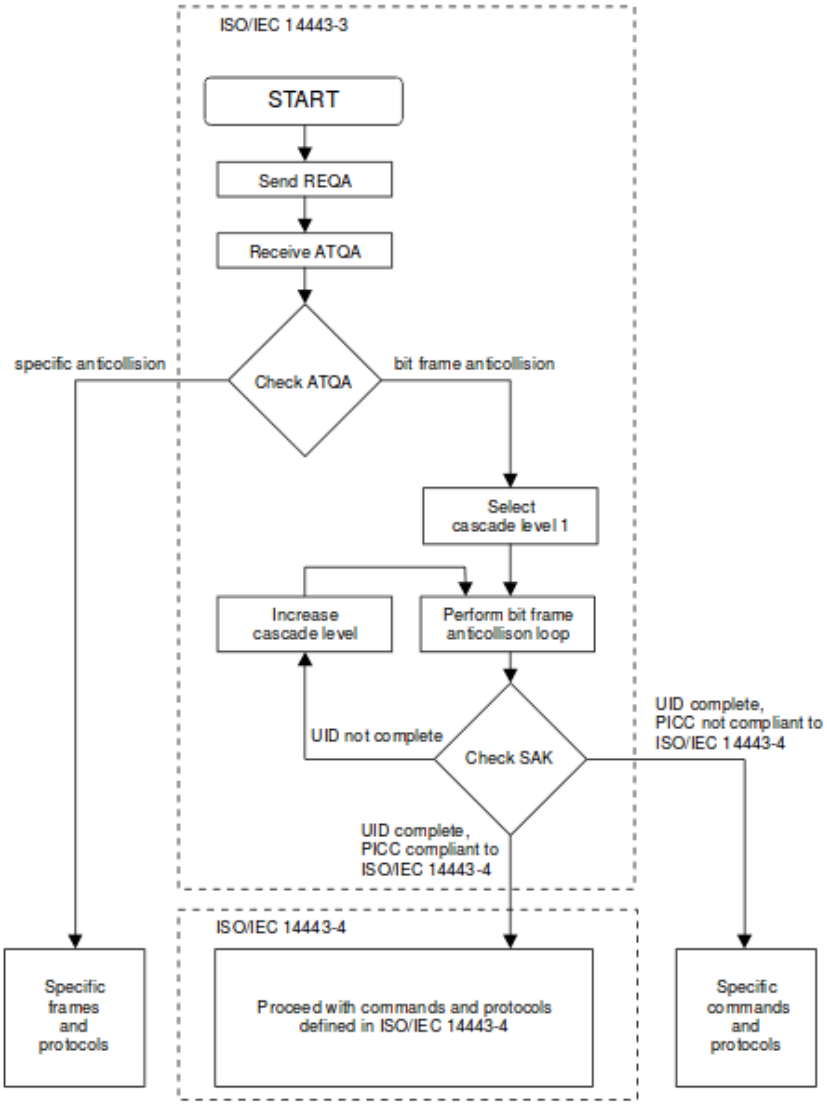

Figura 13. Flujo de Inicialización y anticolisión del PCD (Standardization, 2001b).

En la figura 14 se muestra el bucle anticolisión:

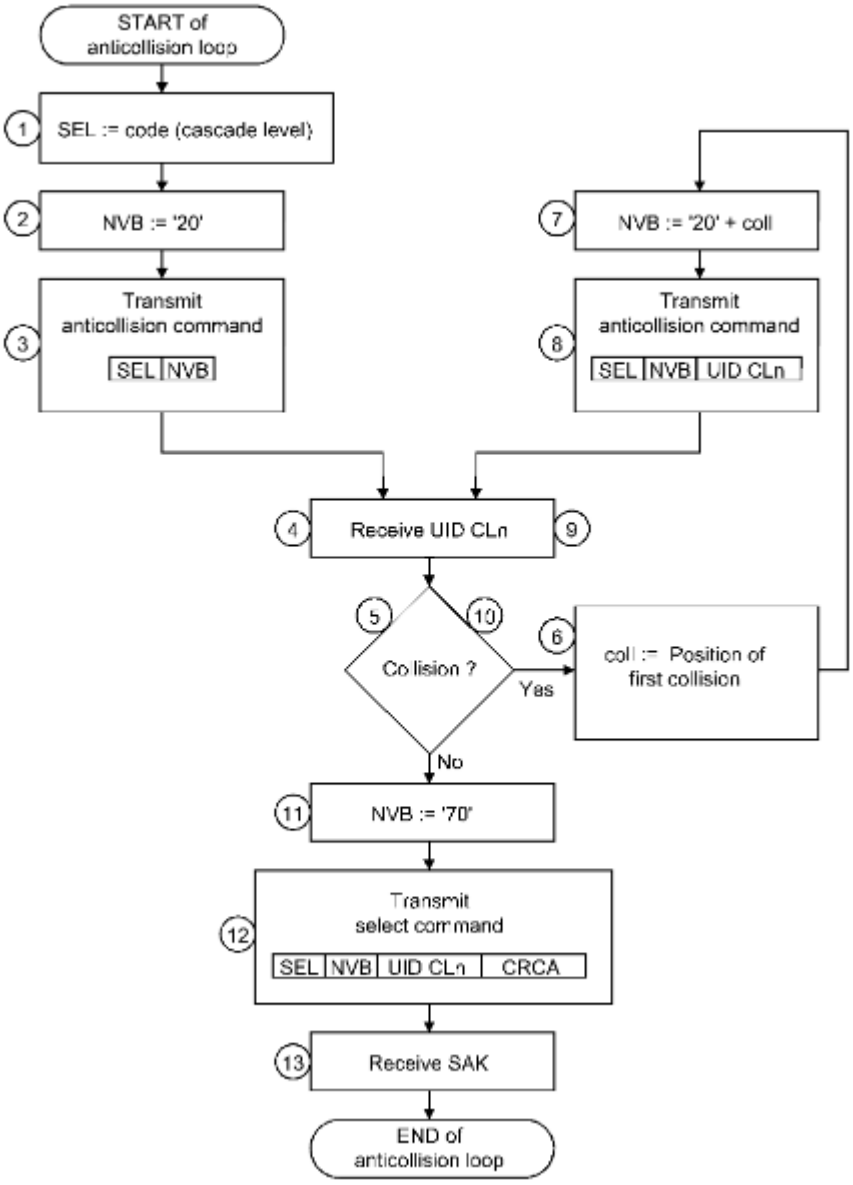

Figura 14. Bucle anticolisión para el PCD (Standardization, 2001b).

Código de selección de acuse de recibido SAK (Select Acknowledge): El código SAK es transmitido por la PICC cuando el NVB tiene especificados 40 bits y cuando todos estos datos concuerdan con el UID CLn. El código SAK, de 1 byte, es transmitido como una trama estándar

seguida de un CRC de dos bytes. El PCD revisa el bit 3 del SAK para conocer si el UID está completo, como se muestra en la tabla 7:

Tabla 7. Código SAK

b8	b7	b6	b5	b4	b3	b2	b1	Significado
x	x	x	x	x	1	x	x	UID incompleto
x	x	1	x	x	0	x	x	UID completo. PICC Compatible con ISO/IEC 14443
x	x	0	x	x	0	x	x	UID Completo. PICC incompatible con ISO/IEC 14443

Si el UID no está completo el PICC deberá volver al estado READY y ejecutar de nuevo el bucle anticolisión incrementando el nivel de cascada CL.

Contenidos UID y niveles de cascada CL: El UID consta de 4, 7 o 10 UID bytes. En la Tabla 9 se muestra la relación entre el nivel de cascada CL y el tamaño del UID.

Tabla 8. Relación de nivel de cascada con tamaño del UID.

Máximo nivel de Cascada	Tamaño del UID	Numero de UID bytes
1	Sencillo	4
2	Doble	7
3	Triple	10

El contenido del identificador UID se describe a continuación (Standardization, 2001b):

UID CLn: Parte del UID de acuerdo al nivel de Cascada, $3 \geq n \geq 1$

UIDn: Byte #n del UID, se envían de a 5 bytes, $n \geq 0$

BCC: UID CLn, es calculado como una XOR con los 4 bytes previos.

CT: Cascade tag, ‘88’

El primer byte del UID (uid0) define el contenido de los siguientes bytes, como se muestra en la tabla 9:

Tabla 9. Descripción del byte uid0.

Uid0	Descripción
0x08	uid1 al uid3 son números generados aleatoriamente
0x0 - 0x7	Números fijos
0x18 – 0xF8 - 0xF	RFU

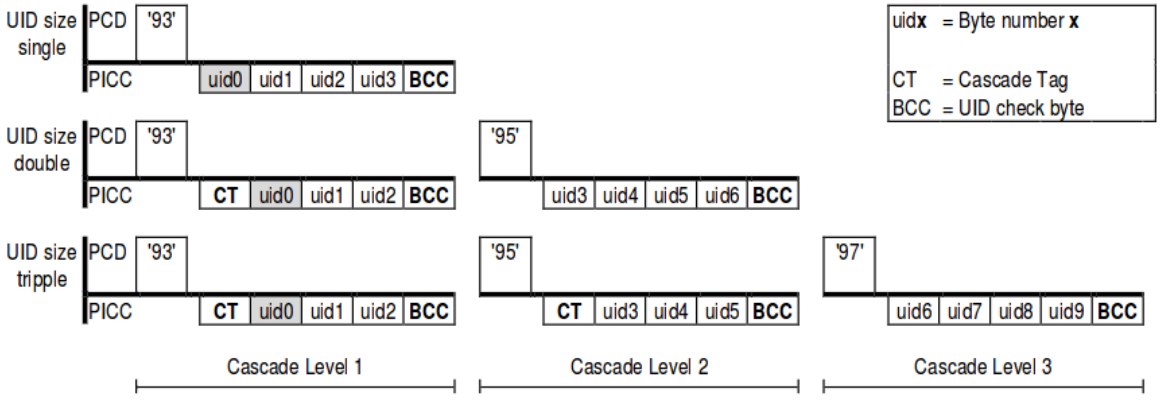

Figura 15. Uso de los niveles de cascada.

Parte 4: Protocolos de transmisión

Protocolo de activación del PICC: Se debe aplicar la siguiente secuencia para activar la comunicación con la PICC (Standardization, 2001c):

- Se inicia la secuencia definida en ISO/IEC 14443-3 (REQA, Bucle de Anticolisión y SEL)
- Se debe revisar la trama SAK para ver la disponibilidad de una respuesta de selección (ATS, Answer to Select).
- La PICC podría estar en el estado HALT si el ATS no está disponible.
- EL RATS (Request for Answer To Select) puede ser enviado por el PCD como un comando después de recibir el SAK si un ATS está disponible.
- La PICC deberá enviar un ATS como respuesta a la respuesta del RATS. La PICC deberá responder el RATS solo si el RAST es recibido después de la selección.

- Si la PICC soporta cambiar algunos parámetros en el ATS, un selector de protocolos y parámetros (PPS, Protocol and Parameters Selection) puede ser usado por el PCD como el siguiente comando después de recibir el ATS para cambiar los parámetros.
- LA PICC enviara una respuesta PPS para responder la solicitud PPS.

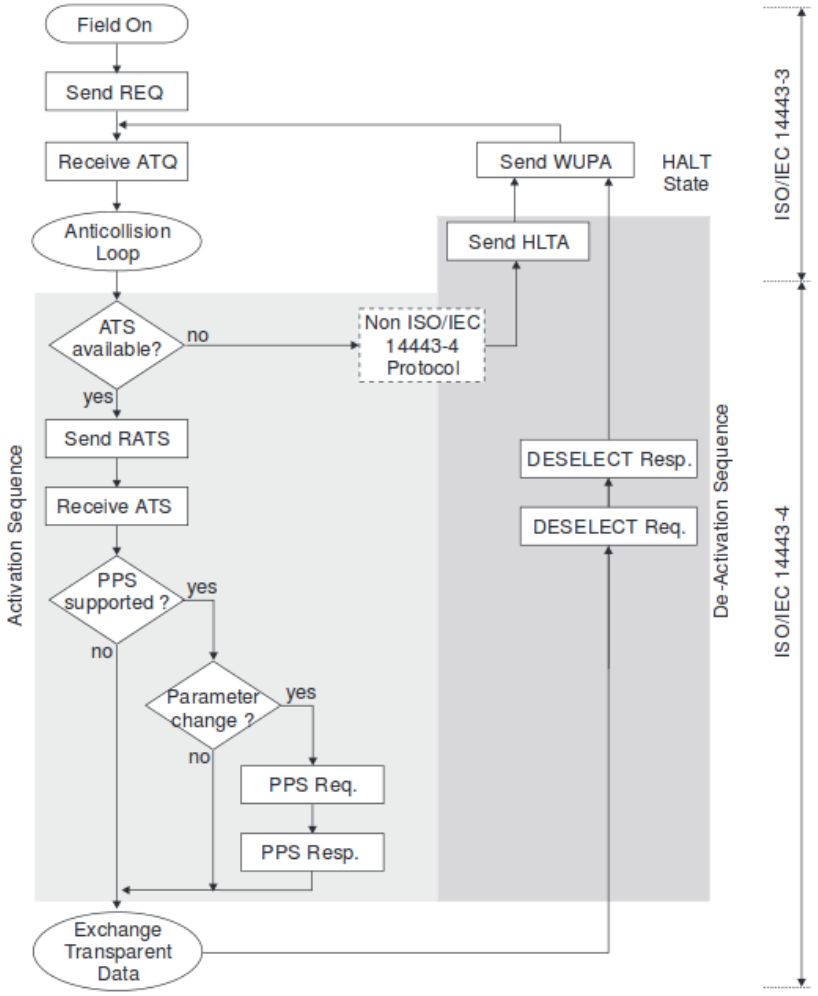

Figura 16. Activación de un PICC por un PCD (Standardization, 2001c).

Solicitud de respuesta para seleccionar (RATS, Request for Answer To Select): La trama RATS se define como se muestra en la siguiente tabla:

Tabla 10. Trama RATS.

'E0'	Parámetro	CRC
1 byte	1 byte	2 bytes

El byte Parámetro consta de dos partes:

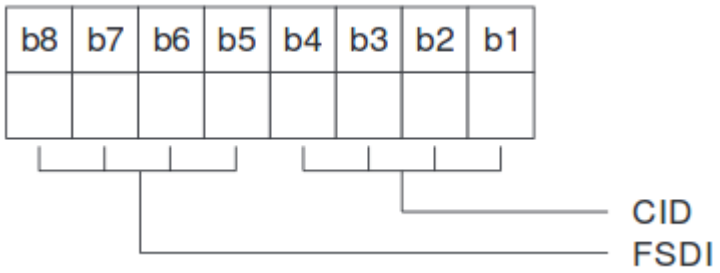

Figura 17. Código del byte Parámetro del RATS (Standardization, 2001c).

Los 4 primeros bits más significativos son llamados Tamaño de Trama para dispositivo de Acoplamiento de Proximidad Entero (FSDI, Frame Size for Proximity Coupling Device Integer) y códigos FSD (Frame Size for Proximity Coupling Device). El FSD define el tamaño máximo de una trama que el PCD es capaz de recibir. En la tabla 12 se muestra la conversión de FSD y FSDI.

Tabla 11. Conversión FSD y FSDI.

FSDI	'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'-'F'
FSD	16	24	32	40	48	64	96	128	256	RFU

Los 4 bits menos significativos son llamados Card Identifier (CID) y definen el número lógico para direccionar el PICC en un rango de 0 a 14. El CID es especificado por el PCD y debe ser único para todos los PICC que estén activos al mismo tiempo.

Answer To Select (ATS): La siguiente figura define la estructura de un ATS:

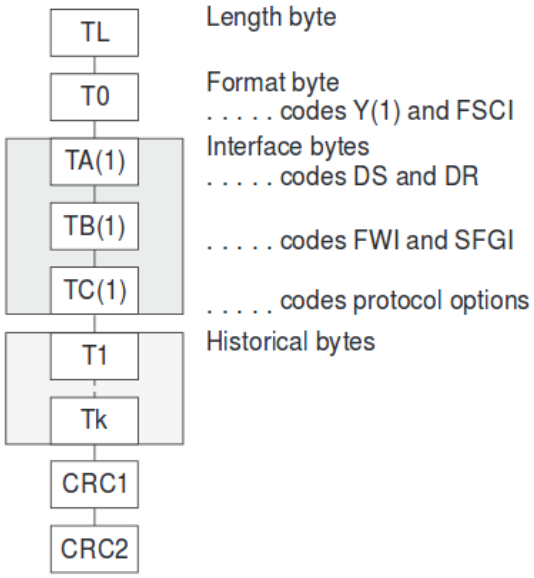

Figura 18. Estructura ATS (Standardization, 2001c).

Byte de Longitud (Length byte): Este campo especifica el tamaño del mensaje ATS sin incluir los códigos de redundancia cíclica (CRC). Este tamaño no debe superar el indicado por el FSD.

Byte de Formato (Format byte): El formato de bytes T0 es opcional y está presente cuando la longitud es mayor que 1. El ATS sólo puede contener los siguientes bytes opcionales, cuando este byte de formato está presente.

- El más significativo b8 se establece en 0. El valor 1 es RFU.
- Los bits b7 a b5 indican la presencia de interfaz posterior bytes de TA (1), TB (1) y TC (1).
- Los bits menos significativos de b4 a b1 se llaman Frame Size for proximity Card Integer (FSCI) y códigos Frame Size for proximity Card (FSC). El FSC define el tamaño máximo de una trama aceptada por el PICC. El valor por defecto de FSCI es 2 y conduce a un FSC de 32 bytes. La codificación de FSC es igual a la codificación de FSD.

Bytes Históricos (Historical bytes): Los bytes históricos T1 a Tk son opcionales y designan información general. La longitud máxima del ATS proporciona el máximo número posible de bytes históricos. ISO / IEC 7816-4 especifica el contenido de los bytes históricos.

Protocolos de comando

Son utilizados varios protocolos de comando, basados en dos especificaciones. La especificación original RFID sobre la lectura y escritura de las etiquetas NFC está construida bajo ISO-14443A. Son compatibles bajo este estándar los siguientes protocolos: Philips/NXP Semiconductors Mifare Classic, Mifare Ultralight y NXP DESFire. El intercambio punto a punto está construido sobre el protocolo de control ISO-18092.

La gran diferencia en este punto entre RFID y NFC es el modo de comunicación punto a punto, el cual es implementado utilizando el estándar ISO-18092. Existen dos protocolos: un protocolo de control de enlace lógico (Logical Link Control Protocol, LLCP) y un protocolo de intercambio simple NDEF (Simple NDEF Exchange Protocol, SNEP) que administran los intercambios punto a punto.

NDEF (NFC Data Exchange Format)

El formato de intercambio de datos NFC (NFC Data Exchange Format, NDEF), define un intercambio de datos por medio de mensajes, los cuales están compuestos de registros NDEF.

Este protocolo hace posible que el código de la aplicación se comunique con los datos de lectura y escritura de las etiquetas NFC, los intercambios punto a punto y la emulación de la tarjeta (Bash, 2015)**.**

Modos de operación

Existen tres tipos de operación NFC: lectura-escritura, punto a punto y emulación de tarjeta. Cada uno tiene sus propias características, aunque es posible definir un modelo de uso para cada modo, que contenga las características obligatorias de cada uno.

Modo de operación lectura – escritura

Se refiere a la comunicación que se presenta entre un teléfono móvil compatible con NFC y una etiqueta NFC con el fin de escribir datos o leerlos sobre éstas. Define dos modos diferentes: el modo lectura y el modo escritura:

En el modo lectura, el iniciador lee los datos de la etiqueta generando un campo magnético a 13.56MHz.

En el modo escritura, el teléfono móvil actúa como el iniciador y escribe datos en la etiqueta. Si la etiqueta ya contiene datos previos al proceso de escritura, estos son sobrescritos. Aunque no es una opción común, un lector NFC también puede ser utilizado para leer datos de una etiqueta, o incluso escribirlos. La tasa de transferencia de datos disponible para este modo es 106kbps.

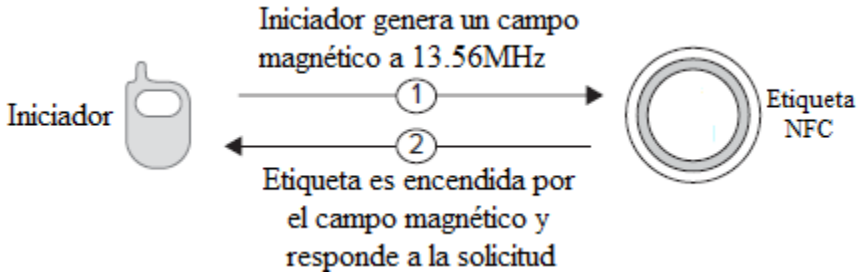

Figura 19. Representación del modo lectura – escritura (Coskun et al., 2013)

Modo de operación punto a punto

Permite utilizar dos dispositivos móviles compatibles con NFC para intercambiar información. Posee dos modos estándares: NFC Interface and Protocol-1 (NFCIP-1) y Protocolo de Control de Enlace Lógico (LLCP).

El protocolo NFCIP-1 aprovecha el hecho de que los dispositivos de destino son definidos previos a comenzar la comunicación, mientras que los dispositivos son idénticos en la comunicación LLCP. Debido a la energía integrada en los teléfonos móviles, ambos se encuentran en modo activo durante este modo de comunicación. Los datos son enviados a través de un canal bidireccional half-duplex, lo que significa que cuando un dispositivo se encuentra transmitiendo, el otro se encuentra escuchando y debe iniciar a transmitir cuando el primero termine. En este modo las tasas de transferencia de datos permitidas son 106, 212 y 424 Kbps.

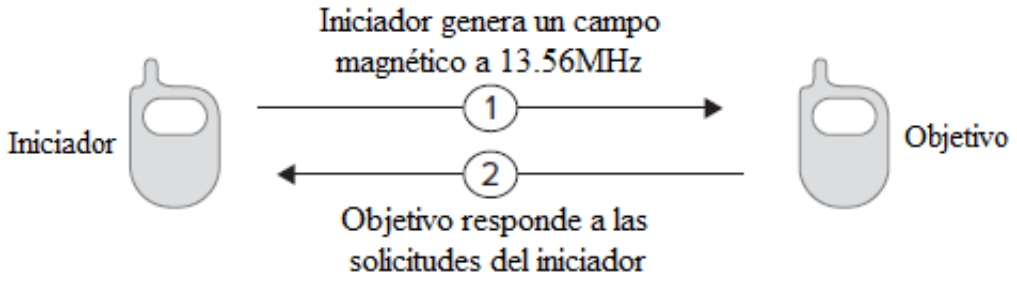

Figura 20. Representación del modo punto a punto (Coskun et al., 2013)

Modo de operación emulación de tarjeta

Ofrece la posibilidad para que un dispositivo móvil compatible con NFC funcione como una tarjeta inteligente sin contacto. El dispositivo móvil no genera su propio campo de radiofrecuencia, sino es el lector NFC el que lo crea. Las interfaces de comunicación soportadas para este modo son: ISO/IEC 14443 Tipo A y Tipo B, y FeliCa.

Este modo tiene gran importancia porque permite el uso de las aplicaciones de pago, además que es compatible con la actual infraestructura de tarjetas inteligentes.

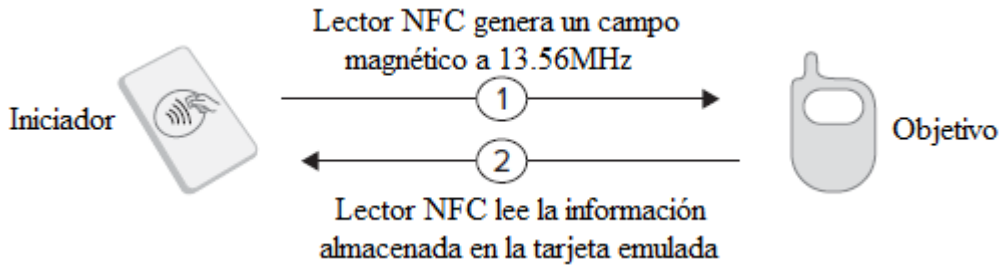

Figura 21. Representación del modo emulación de tarjeta (Coskun et al., 2013)

Aplicaciones NFC

Cada uno de los modos de operación de la tecnología NFC, puede ser implementado en una gran variedad de aplicaciones, permitiendo al usuario una experiencia más natural con la tecnología en las situaciones cotidianas:

Transporte público

El transporte público es uno de los campos donde ha tenido mayor acogida el uso de la tecnología NFC. En tema de acceso a la información, podemos utilizar el modo de emulación de tarjeta de NFC para utilizar nuestro teléfono móvil con el fin de realizar consultas sobre el saldo de viajes, verificar rutas y horarios disponibles, así como recargar cierta cantidad de saldo en nuestra tarjeta virtual (Dunant, 2013).

También podemos usar NFC para interactuar con "Smart Posters" en las paradas de buses, que promocionan eventos, acceso a un mapa, información cultural y turística, entre otros.

Intercambio de datos

Un caso importante del uso de la tecnología NFC en modo punto a punto es el intercambio de datos. Permite a los usuarios mantener altos niveles de seguridad para compartir información privada. Del mismo modo, es posible la transferencia de dinero entre tarjetas virtuales almacenadas en sus dispositivos móviles.

En otros campos como el de las redes sociales, las personas pueden intercambiar información personal con sus amistades. Dos dispositivos pueden ser emparejados como es el caso de los auriculares y los micrófonos, los dispositivos en los autos o los periféricos de los computadores personales.

Entornos inteligentes

Al referirnos a la tecnología NFC en modo de operación de emulación, una de las principales aplicaciones son los entornos inteligentes (Smart Environments). Mediante la previa configuración de datos en el dispositivo móvil un usuario puede administrar las características de un entorno específico (hogar, oficina, sitios públicos) y ajustarlos a su necesidad. Por ejemplo, el control de apertura de ventanas y puertas, la iluminación, selección de canciones.

Alcances

Con el desarrollo de este proyecto se busca proponer un modelo de gestión para las rutas alimentadoras que operan en la estación Banderas del sistema Transmilenio, soportado en tecnologías inteligentes de transporte. Se elaborará un módulo prototipo que funcionará por medio de tecnología NFC y nos permitirá recopilar la información de la demanda de usuarios.

Este módulo se encontrará conectado a un servidor, al que transmitirá la información recopilada, mientras éste último la alojará en una base de datos, desde donde será administraba y utilizada.

Inicialmente se van a definir los criterios a tener en cuenta y que el gestor de trafico autónomo debe evaluar para administrar las rutas de forma eficiente, todo esto soportado en sistemas inteligentes de transporte.

Cada usuario del sistema, para ingresar al área de espera de la ruta alimentadora seleccionada, deberá acercar su tarjeta de acceso (que contendrá embebida la etiqueta NFC) al dispositivo lector del gestor de tráfico, con el fin de registrar su ruta de destino y hora de ingreso. Esta información será guardada en la base de datos y comparada con la información preestablecida de demanda para la ruta y hora correspondiente, con el fin de que el software de gestión pueda tomar decisiones sobre el tiempo que le tomará responder a la solicitud.

Cada tarjeta de acceso que porten los usuarios del sistema deberá llevar embebida una etiqueta NFC compatible con el dispositivo lector. La etiqueta y el lector interactuarán en el modo de operación lectura-escritura.

Se realizará un modelo de simulación de las rutas alimentadoras, con el fin de establecer cuáles son las deficiencias y factores que afectan actualmente el sistema en términos de eficiencia ante la demanda de pasajeros. Se propondrán indicadores que describan el comportamiento del

sistema de rutas alimentadoras. Partiendo de estos indicadores se podrá formular indicadores clave de desempeño para evaluar si la gestión del servicio es coherente con la calidad de servicio deseada.

También, se evalúa la posibilidad de realizar un estudio sobre la capacidad de almacenamiento y procesamiento de datos que debería soportar el centro de gestión de tráfico, además de la infraestructura necesaria, si se quisiera implementar en todo el sistema de buses alimentadores y articulados de la ciudad de Bogotá.

Limitaciones

El proyecto será llevado a cabo en la ciudad de Bogotá. No se pretende diseñar un dispositivo de gestión que responda a la demanda de pasajeros de todo el sistema Transmilenio, pues desarrollar un sistema de esta magnitud constituye un gran manejo de recursos; por tal razón el diseño sólo será aplicado a la plataforma sur de rutas alimentadoras de la estación Banderas. Esto nos permitirá enfatizar en los problemas que surgen sobre este punto del sistema.

Para determinar la reducción de tiempos de permanencia de los usuarios en la plataforma de rutas alimentadoras seleccionada, requeriremos de la cooperación por parte de Transmilenio S.A. y de los mismos usuarios del sistema para obtener la información suficiente, con base en los métodos de búsqueda definidos en encuestas de servicio y estudios del comportamiento del servicio.

El tipo y la capacidad de información obtenida por medio de la lectura de las etiquetas NFC embebidas en las tarjetas de acceso de los usuarios del sistema, estará sujeta a la capacidad de almacenamiento disponible en cada una y a la velocidad de procesamiento del dispositivo electrónico que soportará el módulo de gestión de demanda diseñado.

Las aplicaciones y software utilizado para el desarrollo del proyecto que permitirán integrar la tecnología NFC, estarán enmarcados en el uso de software libre, con el fin de reducir los costos requeridos para llevar a cabo el proyecto.

Metodología

El presente proyecto estará enmarcado bajo un estudio exploratorio de conocimiento, dado las pocas referencias sobre la gestión de demanda en los sistemas de transporte público mediante tecnología NFC. A partir de este estudio, será posible formular nuevas propuestas para aplicar el uso de sistemas inteligentes de transporte en la resolución de las necesidades de movilidad que surgen en grandes ciudades como Bogotá.

El proyecto será desarrollado bajo la metodología experimental, pues buscamos mediante la manipulación de variables como la cantidad de buses y frecuencia de rutas, controlar y gestionar la demanda de usuarios del sistema de transporte, y así ofrecer una alternativa que utiliza sistemas inteligentes de transporte.

Las fases mediante las cuales se ejecutará todo el proyecto, fueron establecidas de la siguiente forma:

Tabla 12. Fases del marco metodológico

FASE	OBJETIVOS	ACTIVIDADES A DESARROLLAR	RESULTADO
Planteamiento y análisis del problema	Diseñar un prototipo para gestionar la demanda usuarios de los buses alimentadores de la estación banderas, perteneciente al sistema Transmilenio en la ciudad de Bogotá D.C, mediante el uso de tecnología NFC.	▪ Efectuar los estudios de los equipos y recursos tecnológicos requeridos para realizar el proyecto. ▪ Realizar el montaje de los equipos, instrumentos y aplicaciones que permiten el funcionamiento del módulo de gestión. ▪ Ejecutar las pruebas y verificaciones al sistema para determinar su viabilidad.	Ofrecer un sistema de gestión alternativo en el sector del transporte público capaz de mejorar la calidad de servicio, con el uso de tecnologías inteligentes.
Recopilación de información	Determinar la demanda de pasajeros que utilizan el servicio de transporte	▪ Investigar mediante los diferentes medios de información al público las estadísticas de la demanda de pasajeros en	Información detallada asociada a la demanda de pasajeros en la

	Transmilenio en las paradas de buses alimentadores de la estación Banderas, para definir la tasa de muestra a emplear en las pruebas del módulo de gestión.	la estación banderas durante los últimos meses. ▪ Realizar visitas de campo para identificar los horarios con mayor flujo de pasajeros.	estación banderas de acuerdo a las rutas alimentadoras y horarios.
Estudio y evaluación de las etiquetas y equipos NFC	Utilizar la tecnología NFC para gestionar el servicio de los buses alimentadores de la estación Banderas.	▪ Determinar el equipamiento y etiquetas NFC disponibles para ejecutar el proyecto. ▪ Evaluar las características de cada elemento y realizar pruebas de funcionamiento. ▪ Valorar los resultados obtenidos en las pruebas y determinar los elementos de mejor desempeño.	Elección de los equipos, instrumentos y etiquetas NFC más convenientes para desarrollar el proyecto.
Montaje del sistema de almacenamiento de datos	Implementar una base datos que contenga la información relacionada a los recursos del sistema (tamaño de la flota, conductores que operan los buses) para optimar su uso.	▪ Evaluar los requerimientos de la base de datos según la recopilación de información previamente efectuada. ▪ Determinar un conjunto de posibles sistemas de gestión de bases de datos que cumplan los requerimientos. ▪ Seleccionar el sistema de gestión que cumpla con las mejores condiciones. ▪ Realizar la instalación, configuración e implementación de la base de datos.	Base de datos con la información de la demanda de pasajeros en funcionamiento.
Desarrollo de la aplicación de usuario	Desarrollo de una aplicación que permita integrar la tecnología NFC con la información almacenada en la base de datos.	▪ Realizar un listado de software libre para desarrollar la aplicación. ▪ Determinar las características y ventajas de cada software. ▪ Seleccionar el software o	Aplicación de software compatible con la tecnología NFC conectada al sistema de gestión de base

		conjunto de software más adecuado para desarrollar la aplicación que permita integrar la tecnología NFC con el sistema de gestión de base de datos. ▪ Esquematizar y desarrollar la aplicación.	de datos.
Verificación del sistema de gestión y pruebas de funcionamiento	Realizar un sistema de gestión de tráfico robusto y adaptable a los cambios del flujo de información, soportado en ITS.	▪ Efectuar pruebas de funcionamiento y operación del sistema de gestión. ▪ Corregir y modificar los errores en la integración de las aplicaciones, en caso de presentarse.	Integración del sistema de gestión de demanda y el módulo prototipo, debidamente verificado.
Resultados del proyecto y conclusiones	Proponer indicadores clave de desempeño (Key Performance Indicators, KPI) que permitan evaluar la gestión del servicio.		

Cronograma

Tabla 13. Cronograma

No	Actividad	Mes	1				2				3				4			
		Semana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
		Duración (semanas)																
I	**Planteamiento y análisis del problema.**	4																
II	**Recopilación de información.**	3																
III	**Estudio y evaluación de las etiquetas y equipos NFC.**	1																
IV	**Montaje del sistema de almacenamiento de datos.**	3																
V	**Desarrollo de la aplicación de usuario.**	3																
VI	**Verificación del sistema de gestión y pruebas de funcionamiento.**	3																
VII	**Resultados del proyecto y conclusiones.**	3																
IX	**Redacción del documento de trabajo de grado**	16																

Desarrollo del proyecto

De acuerdo a las fases del proyecto descritas en la metodología del proyecto, el proceso llevado a cabo para cumplir con cada una de las actividades planteadas se detalla a continuación:

Recopilación de información

Teniendo en cuenta el objetivo principal planteado para este proyecto, se tomó como caso de estudio la estación Banderas del sistema Transmilenio de la ciudad de Bogotá, y las rutas alimentadoras que allí operan.

Para la recopilación de la información fue necesario delimitar los datos de la muestra representativa para el problema planteado, especificando su tamaño y método de recolección. Con base en el planteamiento de un modelo de gestión de demanda de pasajeros de un sistema de transporte masivo, se tomó como unidad representativa de análisis a la cantidad de pasajeros por unidad de tiempo que son despachados por cada una de las rutas alimentadoras.

La muestra se delimitó para evaluar la cantidad de usuarios que utilizan las rutas alimentadoras del sistema de transporte masivo Transmilenio en la estación Banderas. Esta estación cuenta actualmente con 6 rutas alimentadoras:

- Plataforma Sur
    - Kennedy Central
    - Kennedy Hospital
    - Timiza
    - Corabastos
- Plataforma Norte
    - Castilla
    - Biblioteca El Tintal

Se seleccionaron tres de las rutas alimentadoras anteriormente mencionadas, debido a sus características de flujo de pasajeros durante las horas pico:

- Kennedy Hospital
- Kennedy Central
- Timiza

Para recopilar información sobre la demanda de pasajeros de estas rutas, se solicitó la información al área técnica operacional de Transmilenio, quienes suministraron los aforos realizados durante el año 2015 en la estación Banderas. Estos datos contienen la siguiente información:

- Flota (número de buses por ruta alimentadora)
- Intervalo (frecuencia de operación de buses)
- Oferta (número de pasajeros esperados)
- Buses/hora por ruta alimentadora
- Ciclo (duración del recorrido de cada ruta alimentadora)

Adicional a esta información, fueron entregados por parte de Transmilenio los registros de los despachos de rutas alimentadoras realizados durante el mes de septiembre del año 2015. Para este caso, la información suministrada corresponde a:

- Hora de llegada y salida de cada bus alimentador a lo largo del día
- Número de identificación de cada bus alimentador
- Pasajeros que se suben en cada despacho de bus alimentador

Con el fin de contrastar esta información, también se realizaron visitas de campo a la estación para recopilar información reciente de la demanda de pasajeros en las horas pico de la tarde, para cada ruta alimentadora estudiada. Las visitas fueron realizadas durante el mes de octubre del presente año. En estas visitas, se pudo evidenciar cómo aumenta la cantidad de usuarios en las

áreas de espera de las rutas alimentadoras, según transcurre la tarde e inicia la hora pico, alrededor de las 5pm:

Figura 22. Área de espera - Timiza - Inicio hora pico PM

Con el aumento de usuarios, también se incrementa de manera acelerada la densidad de usuarios percibida, debido al limitado espacio disponible para cada parada dentro de la plataforma y a los largos períodos de tiempo que a veces los usuarios deben esperar por un bus:

Figura 23. Área de espera - Kennedy Hospital - 6pm

Tipología de vehículos por clase de ruta

De acuerdo a la información suministrada por Transmilenio S.A. a través del anexo técnico, la tipología de los vehículos que operan el sistema se define de acuerdo a características como el recorrido de la ruta, las restricciones de infraestructura y la demanda de usuarios. Teniendo esto en cuenta, para la operación de las rutas alimentadoras se emplean las siguientes tipologías:

Tabla 14. Tipología de vehículos rutas alimentadoras (S.A., 2009)

TIPO DE VEHÍCULO	CAPACIDAD DE DISEÑO	INTERVALO MÍNIMO EN HORA PICO	FRECUENCIA MÁXIMA EN HORA PICO	RANGO DE DEMANDA
Busetón	50	4	15	481-750
Padrón	80	3	20	751-1600

Según las tipologías definidas anteriormente para los vehículos que operan en la plataforma, se asocian los niveles de intervalos mínimo y máximo en función de las franjas del día:

Tabla 15. Intervalos máximo y mínimo por período del día (S.A., 2009)

	INTERVALO HORA PICO		**INTERVALO HORA VALLE**		**INTERVALO MADRUGADA**	
Tipo de ruta	Mínimo	Máximo	Mínimo	Máximo	Mínimo	Máximo
Alimentadora	3	8	3	12	15	45

Parámetros operacionales

Con respecto a las rutas alimentadoras de la estación Banderas, Transmilenio ofrece la información de los parámetros operacionales relacionados a las frecuencias y recorridos de cada una, durante el año 2015. En la siguiente tabla se observa la información relacionada a las rutas en consideración: Kennedy Central, Kennedy Hospital y Timiza.

Tabla 16. Parámetros operacionales - Rutas alimentadoras Estación Banderas (S.A., 2016)

Ruta	**Nombre**	**Buses AM**	**Buses Valle**	**Buses PM**	**Despachos**	**Km/Vuelta**	**KM Total**	**Capacidad Bus**
8-1	Kennedy Central	6	4	6	255	4,598	1172	50
8-2	Kennedy Hospital	9	7	9	233	6,997	1630	80
8-6	Timiza	7	6	6	190	9,232	1754	80

Ruta	**Nombre**	**Intervalo pico AM**	**Intervalo Valle**	**Intervalo Pico PM**	**Ciclo AM**	**Ciclo Valle**	**Ciclo PM**
8-1	Kennedy Central	3,30	6,00	3,30	19,8	24	19,8
8-2	Kennedy Hospital	4,00	6,00	4,30	36	42	38,7
8-6	Timiza	7,00	8,00	7,30	49	48	43,8

Indicadores de evaluación

Transmilenio realiza un seguimiento de la operación mediante la medición del índice de Pasajeros por Kilómetro, IPK, que evalúa la relación oferta/demanda del servicio. Para el período comprendido entre abril y diciembre del año 2015, se presentaron los siguientes resultados, enmarcados en la medición de este indicador:

Tabla 17. IPK - Rutas alimentadoras Estación Banderas (S.A., 2016)

Mes del año 2015	Pasajeros transportados	Kilometraje programado	IPK
Abril	1.630.039	277.645	5,87
Mayo	1.607.675	283.100	5,68
Junio	1.464.014	271.788	5,39
Julio	1.693.139	288.098	5,88
Agosto	1.722.851	283.416	6,08
Septiembre	1.769.189	284.068	6,23
Octubre	1.780.223	288.801	6,16
Noviembre	1.626.135	274.133	5,93
Diciembre	1.558.449	278.974	5,59

Aunque el Índice de Pasajeros por Kilómetro permite medir los costos en que incurre el operador en la prestación del servicio, actualmente no se utiliza un indicador que permita medir los costos de viaje de los pasajeros y la calidad del servicio en función de factores como el tiempo invertido por cada usuario en el uso del sistema (tiempo de espera de la ruta, tiempo de recorrido, tiempo de transferencia inter-modal).

Debido a que los buses transitan sobre carriles mixtos, compartiendo la vía con el tráfico vehicular particular, viéndose expuestos a eventos externos como accidentes o congestiones vehiculares, se hace más importante minimizar los costos para el operador y los costos de viaje de

los pasajeros a partir de los eventos que en cierta medida pueden ser controlados dentro de la operación, como los son, la frecuencia de salida de los buses alimentadores y los tiempos de espera promedio de los usuarios de las rutas alimentadores.

Sin embargo, el método de despacho de buses actual no garantiza unos costos de viaje a los pasajeros suficientemente bajos, ya que los intervalos de las rutas alimentadoras están programados por una frecuencia determinada basada en parámetros operacionales aplicados a largo plazo, sin tener en cuenta algunos factores que influyen en la demanda de pasajeros actual de cada ruta alimentadora.

Simulación de la gestión de rutas alimentadoras

Basados en la información obtenida a partir de los datos suministrados por el área técnica de Transmilenio y los datos obtenidos durante las visitas realizadas a la estación, se llevó a cabo la simulación del comportamiento de la demanda de pasajeros en la estación Banderas, haciendo uso del software PTV VISSIM.

PTV Vissim es un programa de simulación microscópica utilizado para modelar operaciones de transporte multimodal, perteneciente a la suite Vision Traffic. Permite simular patrones de tráfico con gran precisión y crear entornos con gran detalle y realismo, ideales para evaluar diferentes escenarios de tráfico antes de su ejecución. Se utilizó la licencia gratuita ofrecida por el fabricante para aplicaciones científicas, que comprende un gran número de características destacadas, que fueron utilizadas en el proyecto.

Datos de aforos de Transmilenio

Para el modelamiento de la operación de las rutas alimentadoras con el software Vissim, se tuvo en cuenta la información entregada por Transmilenio S.A. relacionada con:

- Tipología de los buses para cada ruta alimentadora

- Número de buses asignados a cada ruta
- Intervalos de frecuencia en cada franja del día
- Aforos de pasajeros en la estación

Las siguientes gráficas representan el ingreso de los pasajeros durante la hora pico en las tres rutas alimentadoras evaluadas:

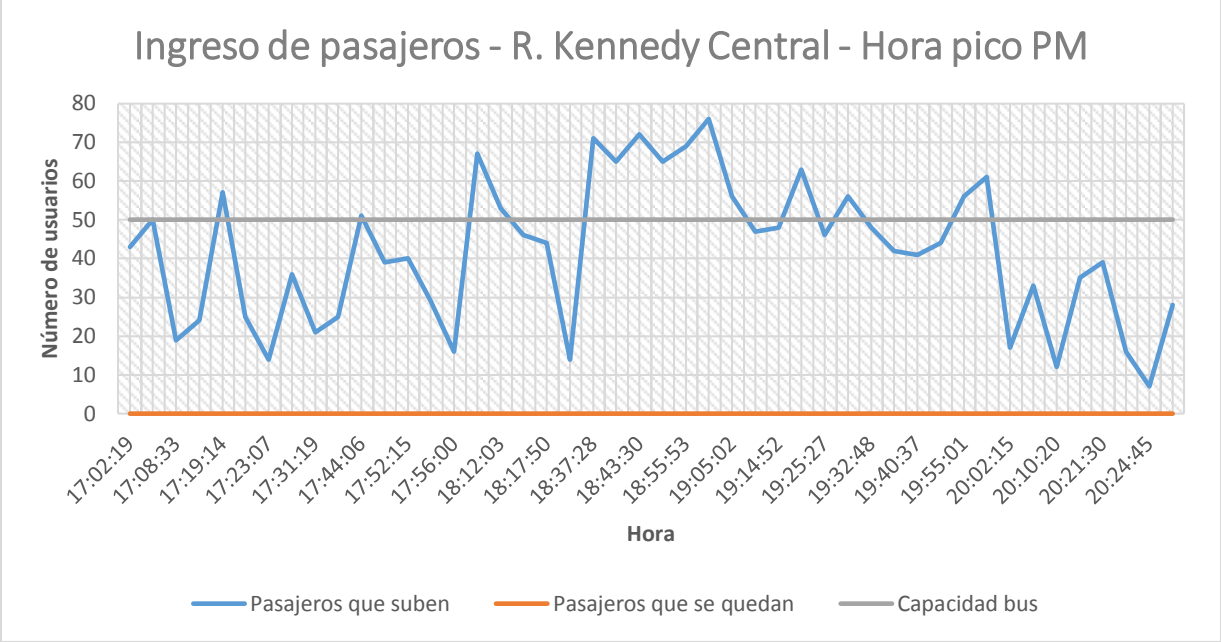

Figura 24. Ingreso de pasajeros - Ruta Kennedy Central - Horario pico PM

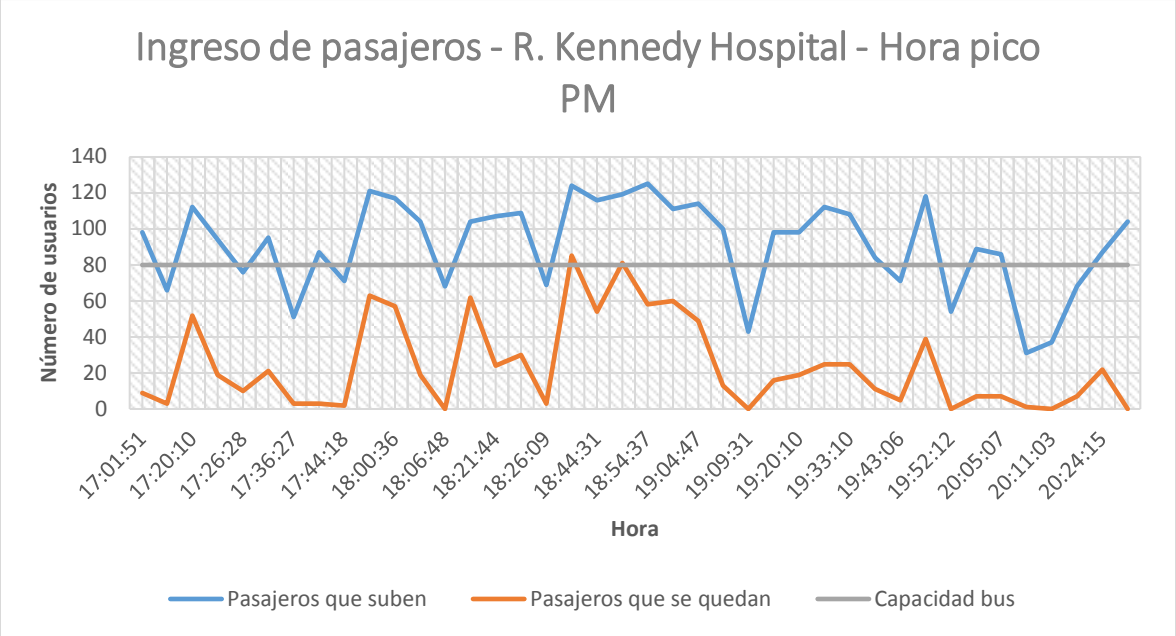

Figura 25. Ingreso de pasajeros - Ruta Kennedy Hospital - Horario pico PM

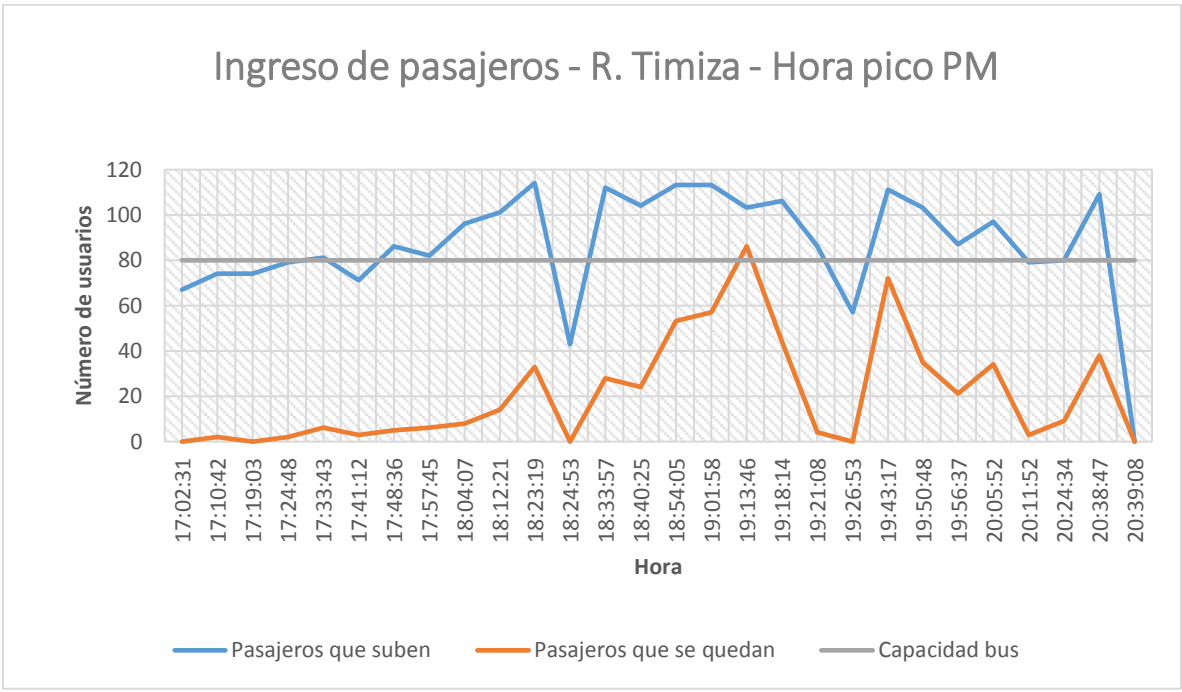

Figura 26. Ingreso de pasajeros - Ruta Timiza - Horario pico PM

El comportamiento del flujo de usuarios mostrado anteriormente permite determinar la correlación entre la demanda y la capacidad de los buses alimentadores y, por ende, la eficiencia de la programación actual de las rutas. Para la ruta Kennedy Central, generalmente los usuarios

que abordan los buses corresponden al total de usuarios que se encuentran en el área de espera, mientras que para las rutas Timiza y Kennedy Hospital la capacidad de los buses generalmente es superada por la gran demanda de usuarios.

Información de visitas de campo

Como se indicó anteriormente, se hizo una recopilación de datos durante las visitas realizadas a las rutas alimentadoras, que nos permitió obtener información del comportamiento de la demanda en la estación, más precisamente en cuanto a la cantidad de pasajeros que esperan por ruta alimentadora, así como la cantidad de pasajeros que abordan cada bus alimentador en las horas pico de los días laborales, los cuales presentan los mayores niveles de congestión del sistema.

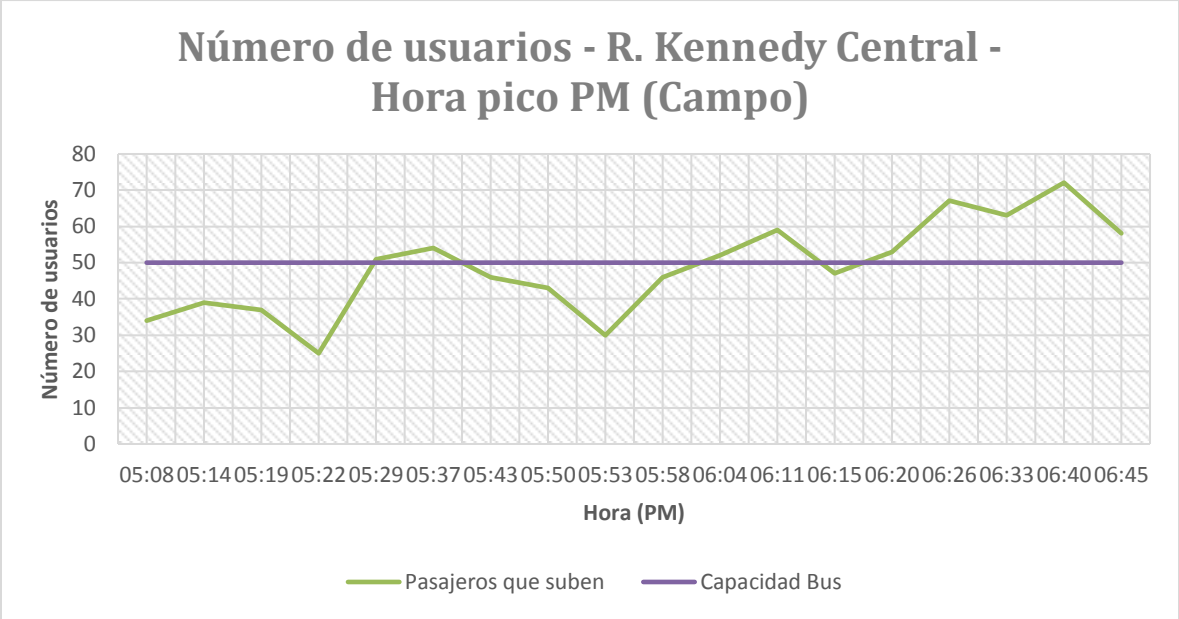

Figura 27. Número de usuarios - Kennedy Central - Hora pico PM (Campo)

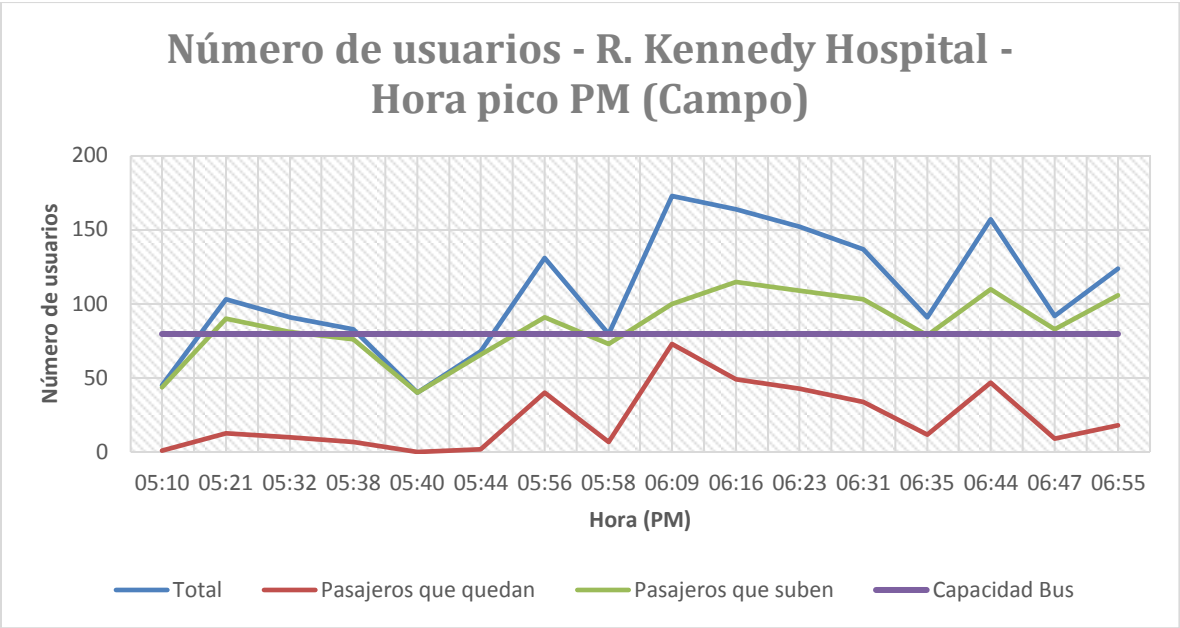

Figura 28. Número de usuarios - Kennedy Hospital - Hora pico PM (Campo)

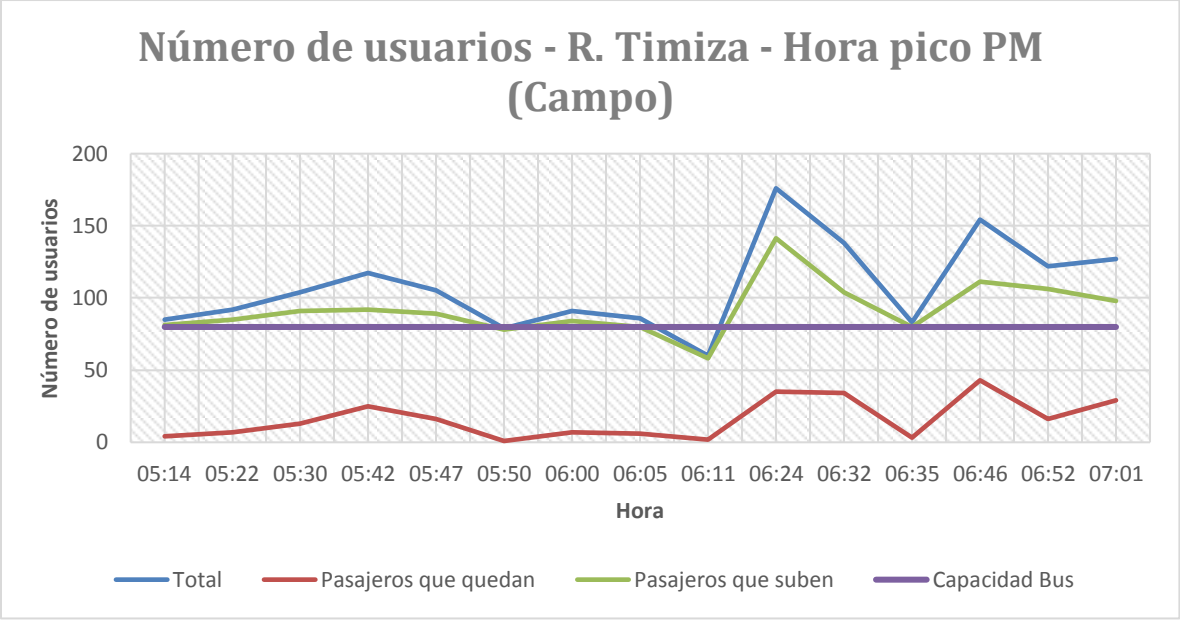

Figura 29. Número de usuarios - Timiza - Hora pico PM (Campo)

Las anteriores gráficas, nos permiten observar que el flujo de pasajeros que recibe cada ruta, sobrepasa en gran parte del tiempo la capacidad nominal de cada bus, incluso si sólo se tienen en cuenta a los usuarios que deben esperar el próximo bus, como se observa en las gráficas de las rutas Kennedy Hospital y Timiza.

Modelo de simulación

Teniendo en cuenta los tiempos de mayor congestión, así como los parámetros operacionales de las rutas alimentadoras, se procedió a realizar la simulación en VISSIM para verificar la capacidad del sistema en función de garantizar bajos costos de viaje a los usuarios y los operadores. En la siguiente figura se observa el escenario general de la estación banderas, diseñado a partir de una imagen satelital:

Figura 30. Vista panorámica de la estación Banderas

El foco de interés de la simulación fue la zona de abordaje de las tres rutas alimentadoras que operan en la plataforma sur de la estación: Kennedy Central, Kennedy Hospital y Timiza. La construcción de las áreas de abordaje fue realizada de acuerdo al área de espera real estimada, el diseño de la vía de los buses alimentadores y al modo de acceso de los usuarios que ingresan desde la plataforma central:

Figura 31. Zonas de abordaje de rutas alimentadoras

Inicialmente, se simuló el escenario actual de despacho de Transmilenio, en el cual las rutas alimentadoras tienen una frecuencia de salida periódica basada en la franja del día, de acuerdo a los estudios de demanda realizados por Transmilenio:

Figura 27. Congestión de usuarios en áreas de espera durante la hora pico

Se tomó como referencia la operación de pasajeros de las horas pico en la tarde con mayor demanda para cada ruta alimentadora, y se muestreó la información en intervalos de 30 segundos.

Resultados de la simulación

La simulación fue configurada para que nos entregara la información asociada al número de pasajeros en cada área de espera, la densidad de pasajeros por área, la densidad promedio experimentada por cada usuario y el tiempo de espera promedio. Estos datos son obtenidos en forma de lista de atributos, los cuales pueden ser guardados en archivos de texto o archivos de bases de datos.

En este caso, la evaluación del modelo simulado se guardó en archivos de texto, los cuales fueron importados a un procesador de hoja de cálculo, para realizar el análisis y visualización de la información obtenida. Los resultados del modelo se generan y visualizan en tiempo de ejecución, lo que permite observar el comportamiento en tiempo real de cada parámetro evaluado. En la siguiente imagen se observa la captura de datos durante la simulación, correspondientes a los atributos de medición de área. Para obtenerlos, se configuró previamente la sección a evaluar en el área de usuarios diseñada, así como la definición de la ruta y entrada de peatones hacia ésta área:

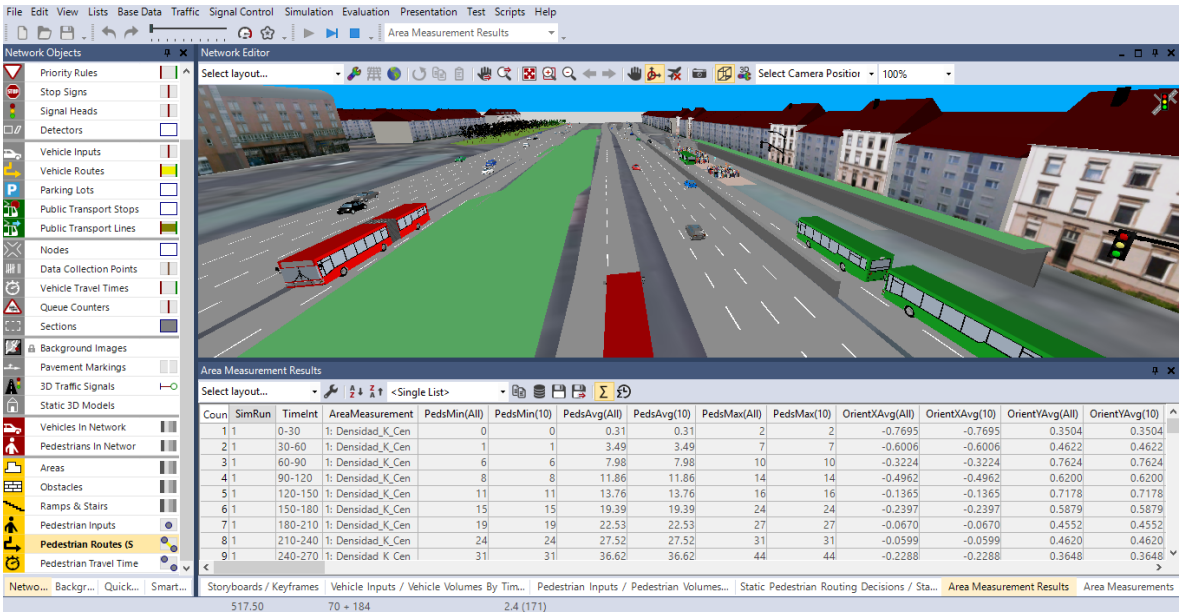

Figura 32. Resultados de la simulación en tiempo de ejecución

En la siguiente imagen se observa la ventana de configuración con la lista de posibles atributos de evaluación, de los cuales son de interés los correspondientes a las mediciones de área:

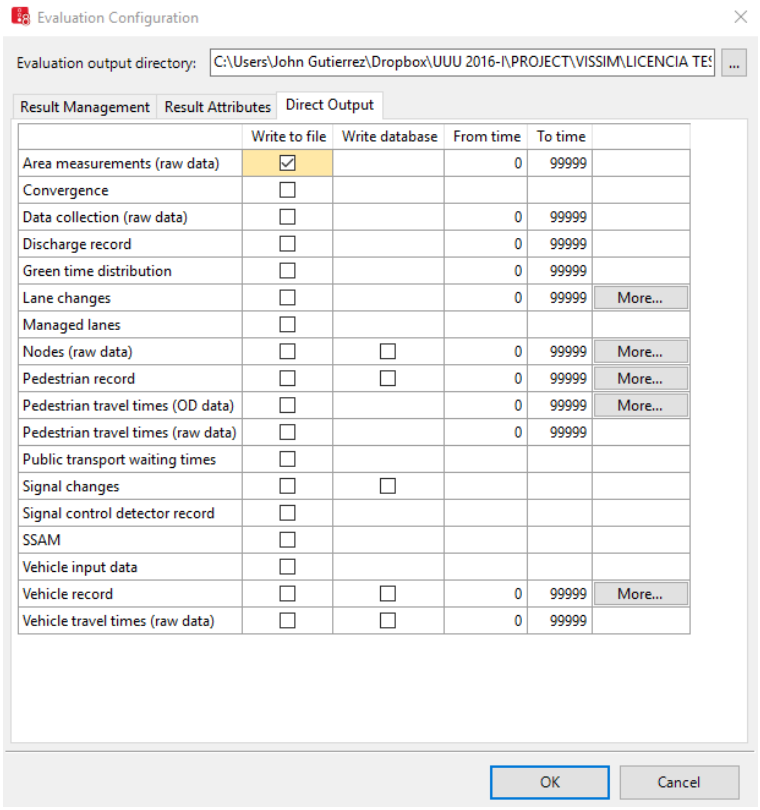

Figura 33. Ventana de configuración de atributos de evaluación

Una vez obtenida e importada la información al procesador de hoja de cálculo, se seleccionó la información obtenida para ser representada gráficamente. Las gráficas del número de usuarios, mostradas a continuación, indican que en la ruta Kennedy Central, la demanda de usuarios es solventada con una programación de rutas periódica, a excepción de dos lapsos de tiempo en los cuales se sobrepasa la capacidad del bus. Incluso, gran parte de los despachos de buses se realizaron con un número bajo de usuarios en relación a su capacidad, generando un costo al operador.

Por otro lado, se observa que las estaciones Kennedy Hospital y Timiza tienen una mayor demanda de pasajeros, generando un incremento constante de la cantidad de usuarios en espera,

en parte debido a que los buses fueron configurados en la simulación para no sobrepasar su capacidad nominal. En este caso, es el costo de viaje de los usuarios es el que se ve afectado, bien sea por la calidad del servicio, como por el tiempo que deben esperar para acceder a un bus.

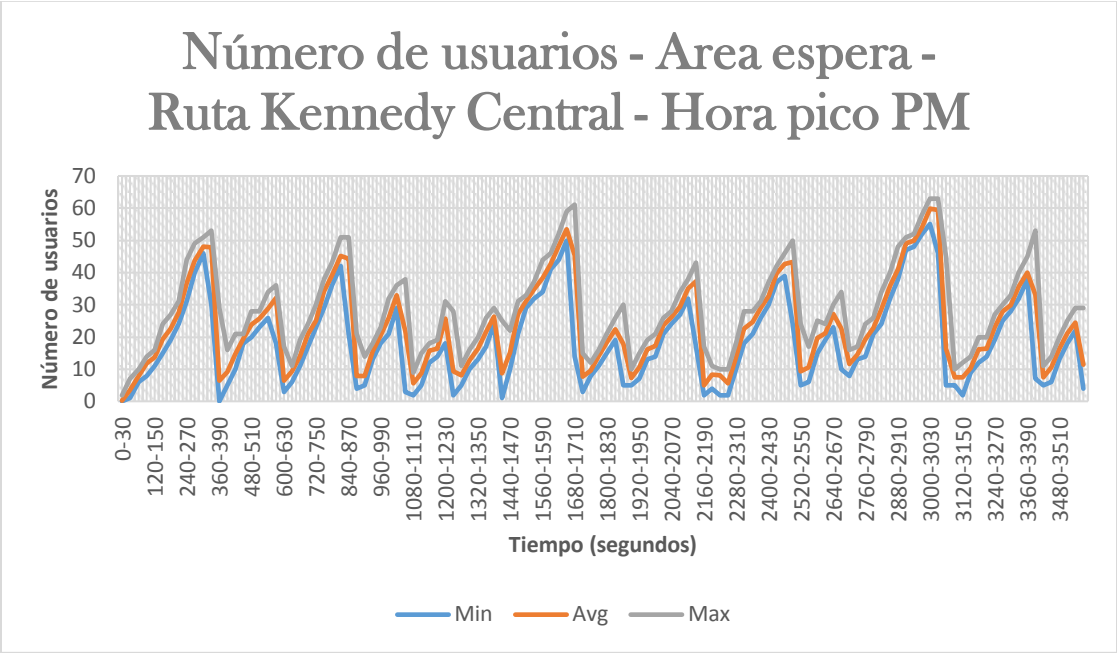

Figura 34. Número de usuarios - Área espera – Kennedy Central – Hora pico PM

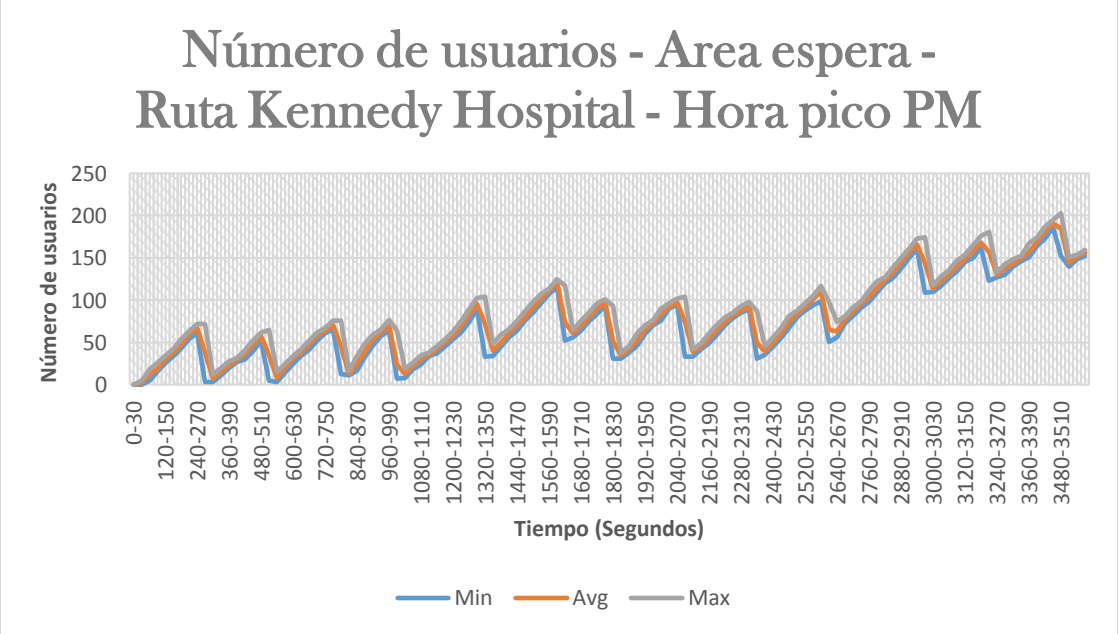

Figura 35. Número de usuarios - Área espera – Kennedy Hospital – Hora pico PM

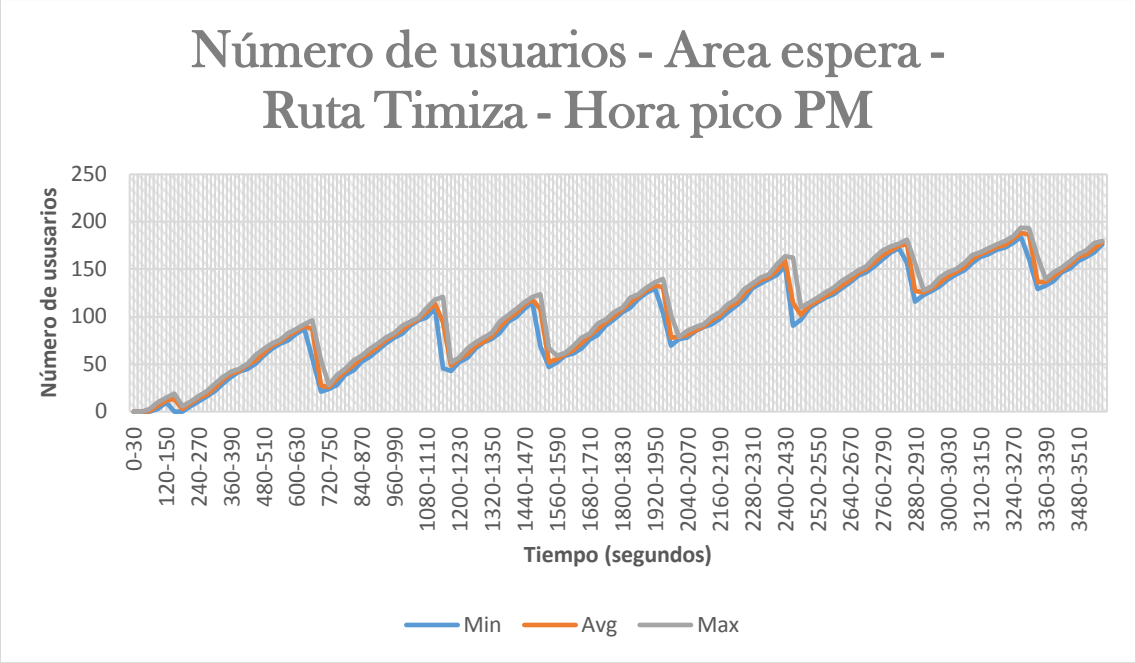

Figura 36. Número de usuarios - Área espera – Timiza – Hora pico PM

Otro atributo evaluado fue la densidad de usuarios, tomando como referencia el área de espera diseñada en la simulación que corresponde de similar forma al área real. Como es de esperar, este indicador varía de manera proporcional al número de usuarios esperando:

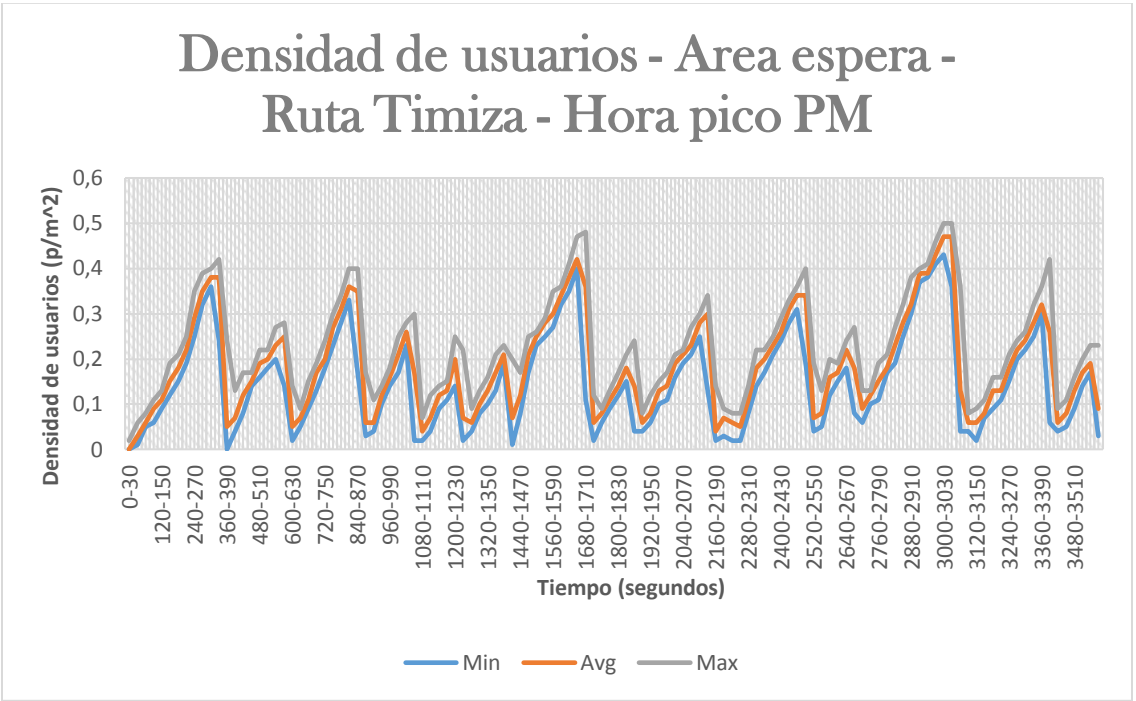

Figura 37. Densidad de usuarios - Área espera – Timiza – Hora pico PM

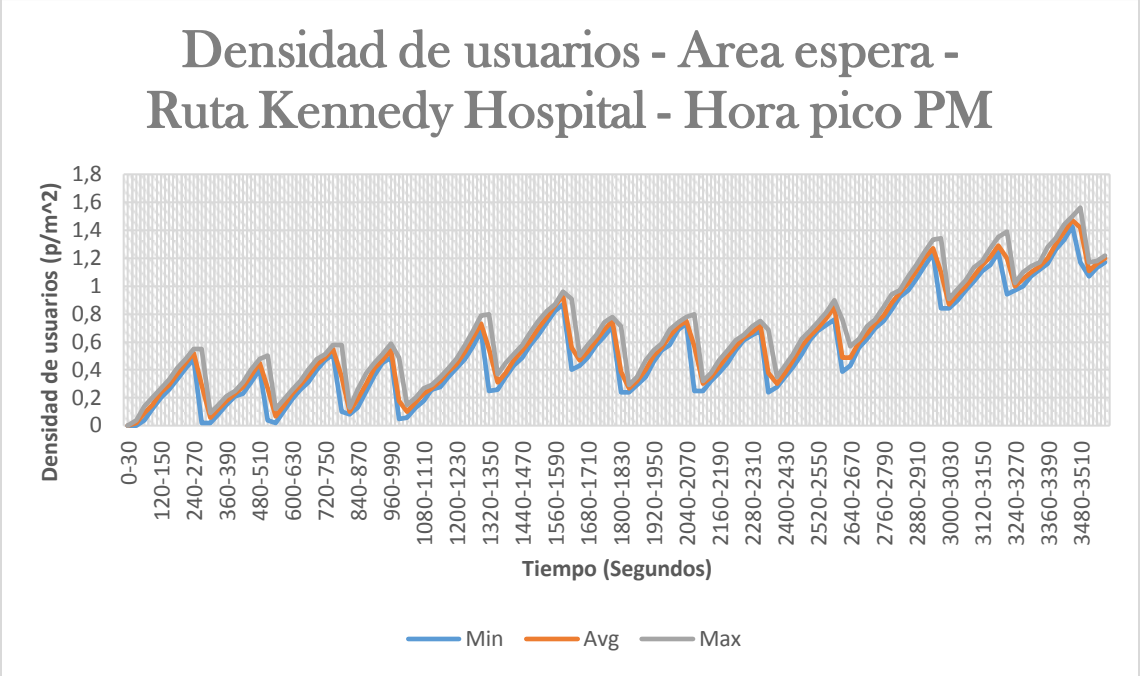

Figura 38. Densidad de usuarios - Área espera – Kennedy Hospital – Hora pico PM

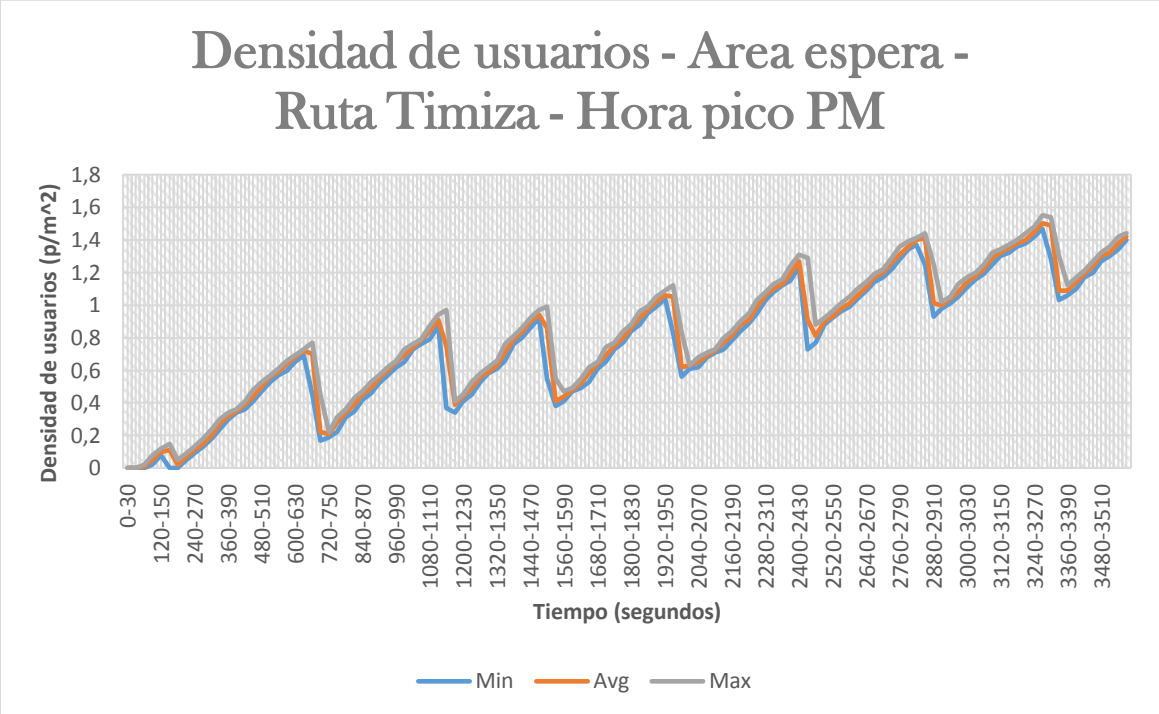

Figura 39. Densidad de usuarios - Área espera – Timiza – Hora pico PM

También se calculó la densidad experimentada dentro del radio de percepción de cada usuario.

Este atributo nos aporta un valor más cercano a la densidad real que perciben los usuarios,

teniendo en cuenta que éstos se ubican en una zona más cercana a las puertas de acceso a los buses; además, indica como los usuarios que se encuentran más cerca de la zona de abordaje experimentan una densidad mucho mayor que aquellos que se encuentran en la zona perimetral del área de espera.

Tomando como referencia los valores máximos de densidad alcanzados, mientras que en la ruta Kennedy Central se observa una densidad de 2 $usuarios/m^2$, concebida como una densidad admisible en términos de comodidad, las rutas Kennedy Hospital y Timiza alcanzan densidades de 3.9 $usuarios/m^2$ y 3.7 $usuarios/m^2$, respectivamente. Esto es un indicador muy probable de hacinamiento, independientemente del tamaño del área.

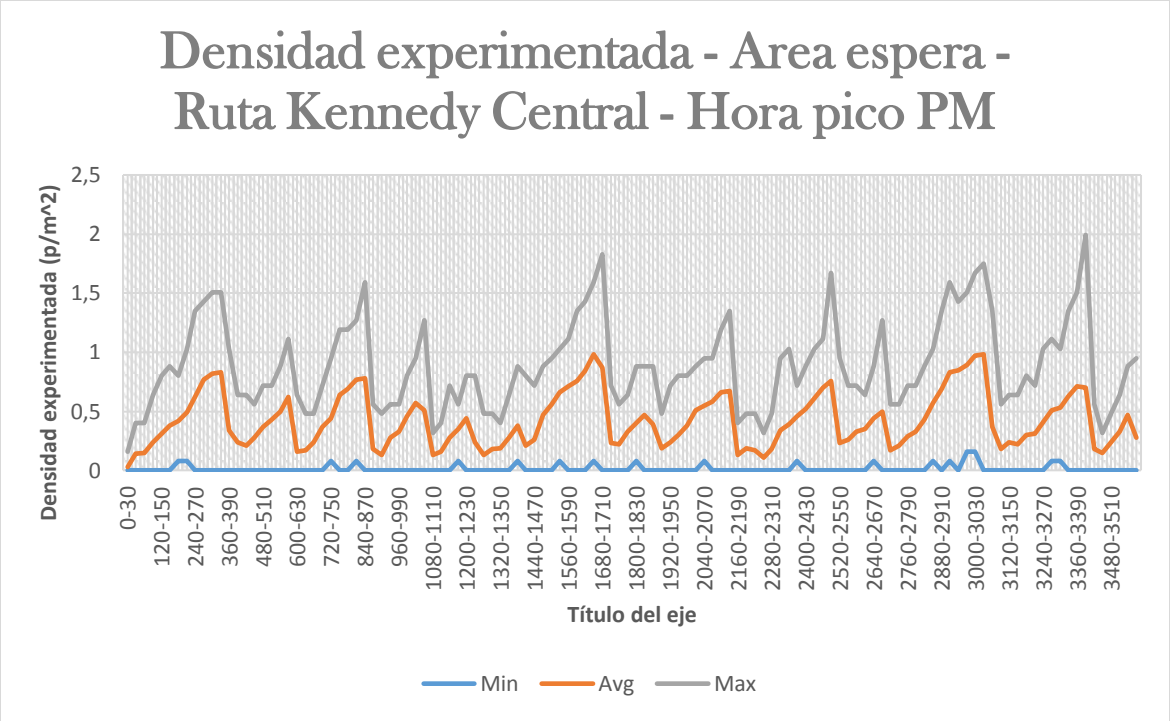

Figura 40. Densidad experimentada - Área espera – Kennedy Central – Hora pico PM

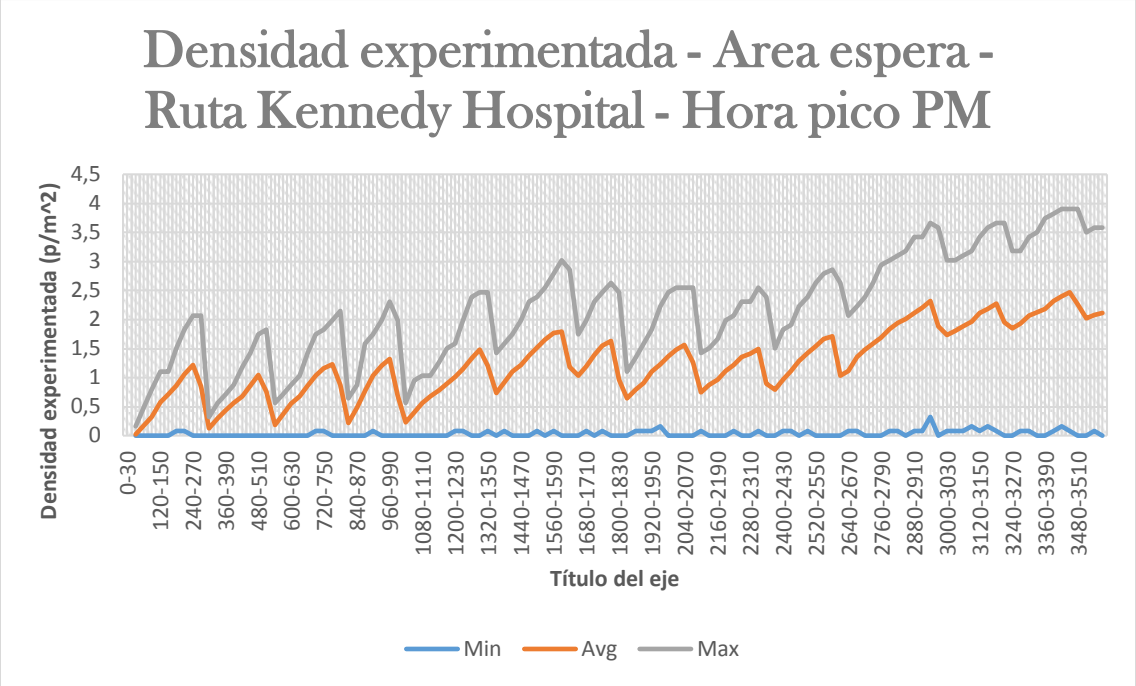

Figura 41. Densidad experimentada - Área espera – Kennedy Hospital – Hora pico PM

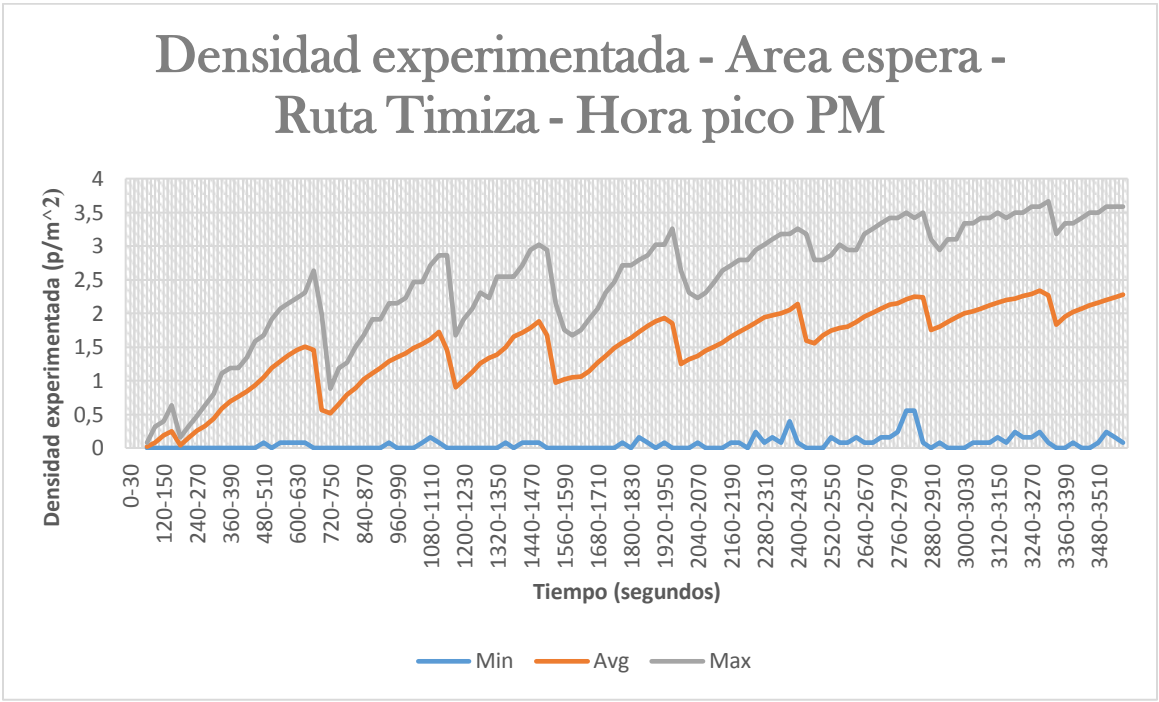

Figura 42. Densidad experimentada - Área espera – Timiza – Hora pico PM

El tiempo de espera se halló midiendo el tiempo que cada usuario dura en el área de espera desde que ingresa a la sección hasta que sale en el bus alimentador. Con base en los intervalos de

tiempo programados para cada ruta, se presentan considerables retrasos en las rutas Kennedy Hospital y Timiza, alcanzando tiempos de espera promedio de 186 segundos y 300 segundos, respectivamente. Esto, teniendo en cuenta el tiempo inicial que la simulación dura el alcanzar el comportamiento real de la plataforma.

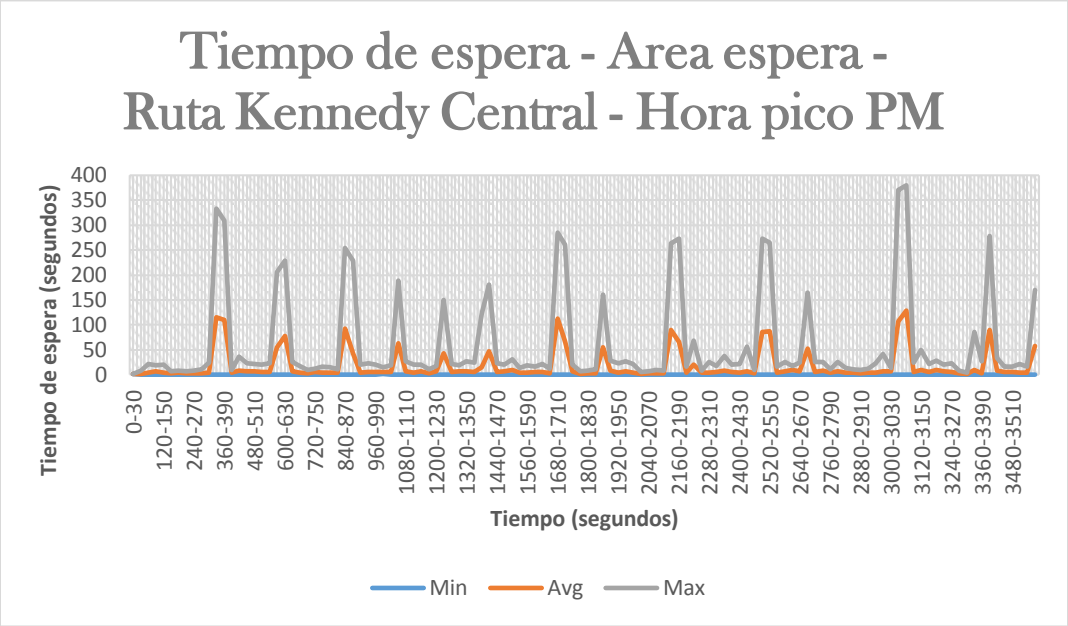

Figura 43. Tiempo de espera - Área espera – Kennedy Central – Hora pico PM

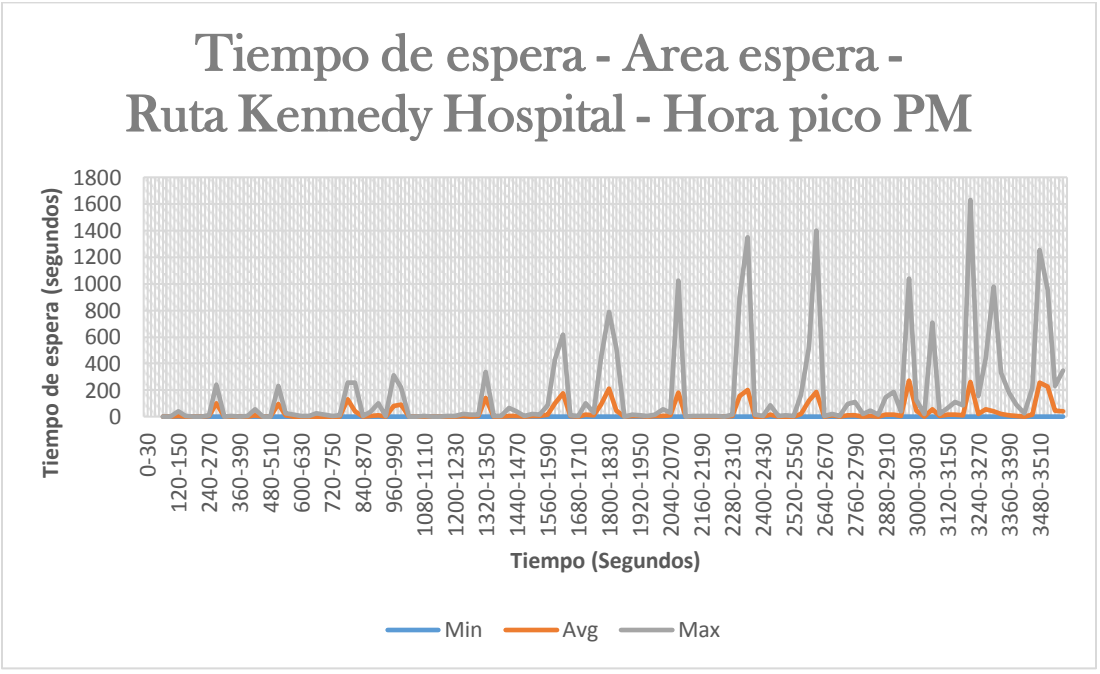

Figura 44. Tiempo de espera - Área espera – Kennedy Hospital – Hora pico PM

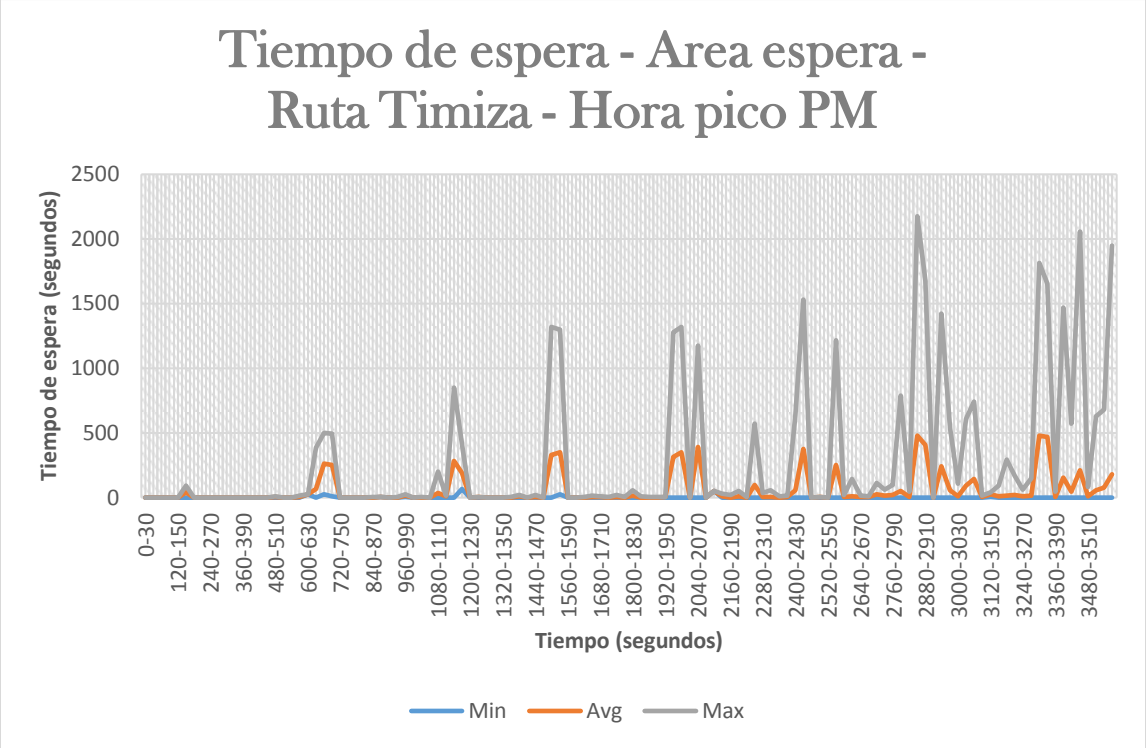

Figura 45. Tiempo de espera - Área espera – Timiza – Hora pico PM

Estudio y evaluación de las etiquetas y equipos NFC

En el grupo de investigación LIDER (Laboratorio de Investigación y Desarrollo en Electrónica y Redes) se cuenta con lectores de tarjetas y tags NFC. De acuerdo con los requerimientos de la unidad de procesamiento de datos, habiendo sido elegida la Raspberry Pi por las razones que se describirán más adelante, se pueden utilizar dos lectores para este propósito: el lector EXPLORE-NFC-WW de la empresa NXP y el RFID 13.56 MHz / NFC Modulo para Arduino y Raspberry Pi.

Pruebas de funcionamiento EXPLORE NFC-WW

Explore NFC es una tarjeta de alto rendimiento para tecnología NFC en modo lector, P2P y emulación, soporta 4 tipos de tags NFC ofrece software de configuración especializado para cada modo NFC de operación.

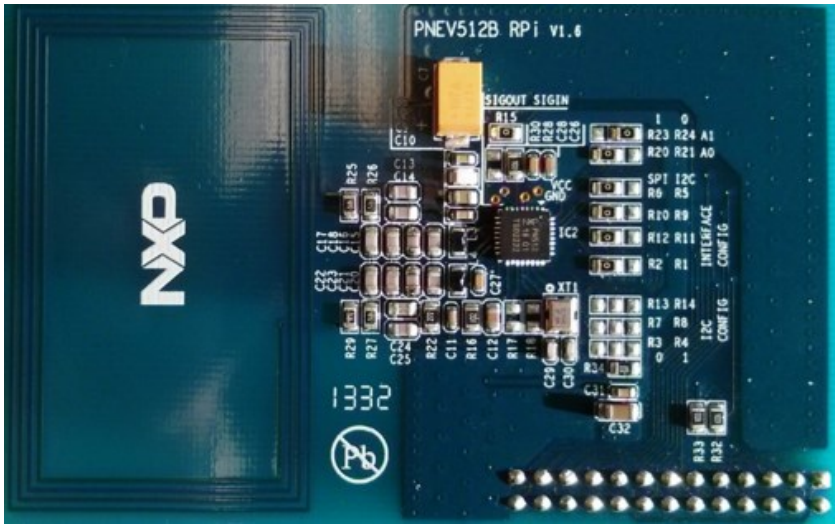

Figura 46. Tarjeta EXPLORE NFC-WW

Para usar el lector EXPLORE-NFC-WW se debe configurar la Raspberry Pi como sigue:

Se realizaron las pruebas de adquisición de datos con el lector. Se hizo uso de una librería desarrollada por la empresa fabricante NXP, basada en la NFC Reader Library NXP, escrita en lenguaje C. Hasta la fecha, solo funciona el protocolo SPI para conectar el modulo lector con la Raspberry Pi con una distribución Linux (Guide, 2015).

Primero, se configuró la Raspberry Pi para activar la Interfaz Periférica Serial (SPI, por su traducción al inglés Serial Peripheral Interface). Para esto se accedió a la configuración de la Raspberry Pi mediante el siguiente comando:

sudo raspi-config

En el siguiente menú, se seleccionó el ítem Advance Options, como muestra la siguiente figura, y posteriormente el ítem SPI. A continuación, se seleccionó la opción <Yes>. En esta parte fue necesario reiniciar el dispositivo mediante el comando:

sudo reboot

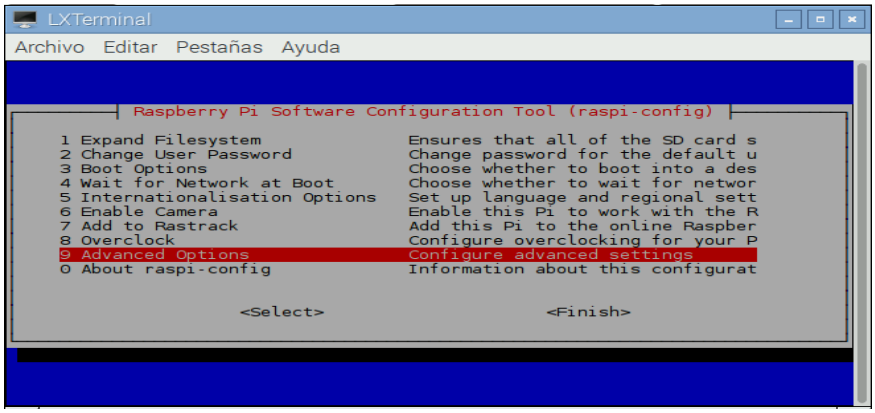

Figura 47. Configuración Raspberry Pi.

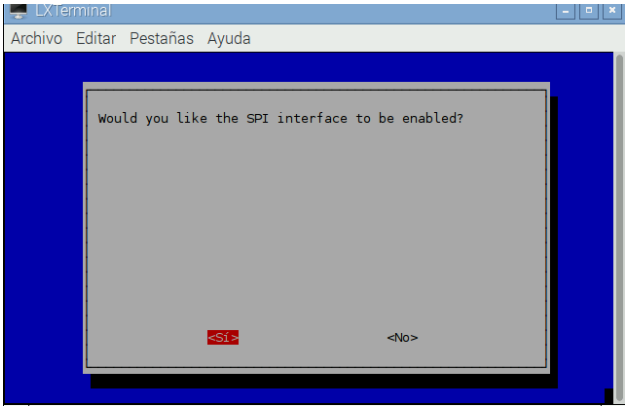

Figura 48. Habilitación de la interfaz SPI.

Para instalar la aplicación EXPLORE-NFC, se descargó el archivo comprimido de la página web de Element L4 Community (Element14, 2016). Con esta aplicación, se realizaron las pruebas iniciales del lector. Se descomprimió el archivo, que contiene tres paquetes *.deb*. Para instalar el software, se debe ejecutar desde el home, o la carpeta donde se aloja el archivo descomprimido, el siguiente comando:

sudo dpkg -i libneardal0_<version>_armhf.deb libwiringpi2-<version>_armhf.deb neard-explorenfc_<version>_armhf.deb

El texto ***<version>*** *fue* reemplazado por los nombres de los paquetes descargados. En nuestro caso, el comando ejecutado fue:

sudo dpkg -i libneardal0_0.14.2-1_armhf.deb libwiringpi2-2.25-1_armhf.deb neard-explorenfc_0.9-1_armhf.deb

Una vez instalado, se validó el funcionamiento del lector con el comando:

explorenfc-basic

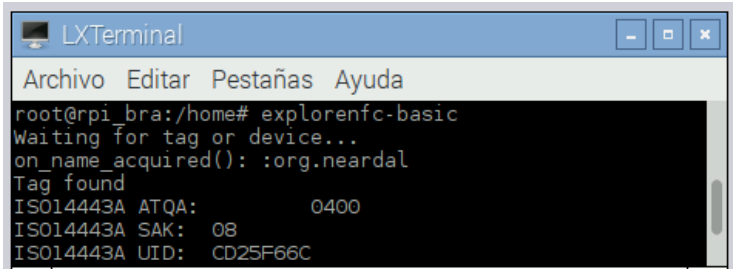

Figura 49. Datos obtenidos con el programa explorenfc-basic.

Como se observa en la figura anterior, se muestra la información: ATQA, SAK y UID. Los datos ATQA y SAK identifican al fabricante de la tarjeta, y el dato UID corresponde al Número de Identificación Única (UID, por su traducción al inglés Unique Identification Number).

También se verificó el ingreso de más datos a las tarjetas o tags, cuya opción Lectura/Escritura se encontraba habilitada, como se muestra en siguiente figura:

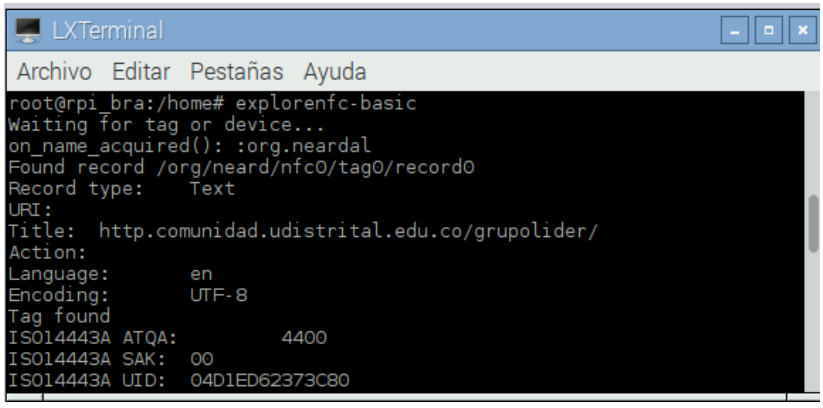

Figura 50. Datos obtenidos de tarjeta con modo Lectura/Escritura.

Pruebas de funcionamiento módulo RFID 13.56 MHz / NFC

El módulo RFID 13.56MHz es compatible en modo lectura/escritura con las normas ISO 14443A, MIFARE, FeliCaTM y NFCIP-1, posee una capacidad máxima de 4KB y permite lectura de tags, llaveros y stickers.

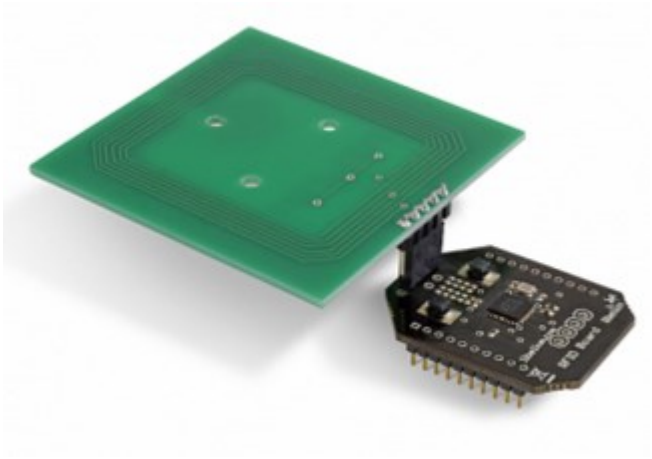

Figura 51. Módulo RFID 13.56 MHz Module for Raspberry Pi

Para realizar las pruebas de adquisición de datos con el lector RFID 13.56 MHz / NFC Modulo para Raspberry Pi, se instaló la librería ArduPi. La librería ArduPi es una librería de C++ que permite escribir programas para Arduino ejecutables en una Raspberry Pi. Fueron de especial interés, las funciones necesarias para la comunicación por el puerto serial, I2C, SPi y GPIO (Hacks, 2016a). Se procedió a instalar la librería ArduPi desde el emulador de terminal con la siguiente línea:

Wget http://www.cooking-hacks.com/media/cooking/images/documentation/raspberry_arduino_shield/raspberrypi.zip && unzip raspberrypi.zip && cd cooking/arduPi && chmod +x install_arduPi && ./install_arduPi && rm install_arduPi && cd ../..

Como se observa, se descargó un archivo desde la página del fabricante que luego se descomprimió en la carpeta ArduPi. Después de realizar la instalación, en la carpeta ArduPi se encontraban 3 archivos: arduPi.cpp, arduPi.h y arduPi_template.cpp. A continuación, se habilitaron las interfaces de comunicación. Para esto, se editó el archivo de configuración:

sudo nano /boot/config.txt

Se agregaron las siguientes líneas de texto para habilitar las interfaces de comunicación:

#enable uart interface

enable_uart=1

#enable spi interface

dtparam=spi=on

#enable i2c interface

dtparam=i2c_arm=on

Posteriormente, se reinició el sistema para actualizar los cambios realizados:

sudo apt-get update

sudo apt-get upgrade

sudo reboot

Luego, se compilaron los programas con g++, el cual fue instalado previamente desde el archivo alojado en la página del fabricante:

wget http://www.cooking-hacks.com/media/cooking/images/documentation/raspberry_arduino_shield/raspberrypi.zip && unzip raspberrypi.zip && cd cooking/arduPi && chmod +x install_arduPi && ./install_arduPi && rm install_arduPi && cd ../..

cd cooking/arduPi

g++ -c arduPi.cpp -o arduPi.o

Los programas creados con extensión .cpp (C++) se compilaron escribiendo la siguiente línea en el emulador de terminal desde la carpeta donde se creó el programa:

g++ -lrt -lpthread MY_PROGRAM.cpp arduPi.o -o MY_PROGRAM

Donde MY_PROGRAM corresponde al nombre del programa. El programa se guardó en la capeta arduPi. Para ejecutar el programa, se accedió a la carpeta arduPi en el terminal y se escribió:

. /MY_PROGRAM

Luego, se probó el lector RFID 13.56 MHz / NFC Modulo para Raspberry Pi con un código diseñado por el fabricante del módulo (Hacks, 2016b). Como se observa en la siguiente figura, se logró acceder al UID y a los datos útiles de las tarjetas.

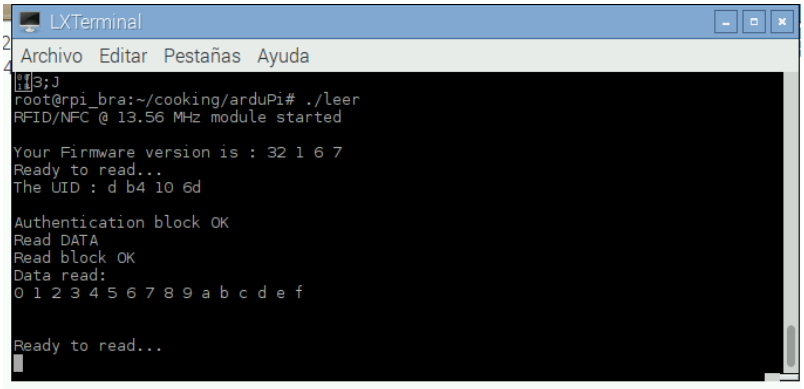

Figura 52. Datos obtenidos con el programa diseñado por el fabricante del módulo EXPLORE-NFC-WW.

Elección del lector NFC

A continuación, se muestran las características técnicas analizadas de los dos lectores probados para definir cuál se iba a utilizar:

Tabla 18. Comparación de lectores NFC

Características Lector	EXPLORE-NFC-WW	RFID 13.56 MHz / NFC Modulo para Raspberry Pi
Distancia Operativa.	Entre 5 y 10 cm	5 cm
Funcionalidad Anticolisión.	ISO/IEC 14443-4 ISO/IEC 14443-3	ISO/IEC 14443-A
Velocidad de Lectura.	424 kbit/s	106 kbit/s
Lenguaje de Programación.	Python, C, C++	Arduino, C++
Tarjetas Compatibles.	FeliCa o ISO/IEC 14443A/MIFARE	ISO 14443A / MIFARE / FeliCaTM / NFCIP-1
Precio.	17,57 euros	50 uros

Características de las tags NFC

Para realizar las pruebas de compatibilidad de las etiquetas y tarjetas disponibles, se tuvieron en cuenta las siguientes referencias disponibles, desarrolladas por el fabricante Mifare:

Mifare Classic EV1

Son tarjetas utilizadas en transporte público, gestión de acceso. Sus características son:

- Compatibilidad con el estándar ISO/IEC 14443A
- Disponible con identificador único de 7 bytes (7B UID) o identificador único de 4 bytes (4B UID)
- Memoria EEPROM de 1KB o 4KB
- Generador de número aleatorio
- Longitud de clave de encriptación 48-bit Crypto-1
- Seguridad de comunicación encriptada
- No es soportado por la mayoría de dispositivos NFC

Mifare Ultralight C

Son etiquetas ideales para implementaciones de bajo costo, aplicaciones de alto volumen como transporte público, sirven como reemplazo de las bandas magnéticas y códigos de barras. Pueden ser fácilmente integrados en esquemas existentes e incluso los equipos expendedores de billetes pueden ser fácilmente actualizados. Sus principales características son:

- Compatibilidad con el estándar ISO/IEC 14443A
- Tasa de transferencia de datos de 106kbit/s
- Anticolisión
- Memoria de 144 bytes
- Criptografía tipo TDES

- Longitud de clave de encriptación de 112bit
- Compatibilidad con dispositivos NFC bajo el NFC Forum tipo 2

Montaje del sistema de almacenamiento de datos

Existe una variedad de gestores de base de datos que pueden ser utilizados para almacenar la información del sistema gestor de tráfico. Por esta razón, se evaluaron las características de cada opción existente, para determinar cuál se ajustaba más a los requerimientos y necesidades de la implementación.

Comparativa de los gestores de base de datos

Existe una variedad de gestores de base de datos que pueden ser utilizados para almacenar la información del sistema gestor de tráfico. Por esta razón, se deben evaluar las características de cada opción existente, para determinar cuál se ajusta más a los requerimientos y necesidades de a la implementación.

SQ-LITE

- Busca proporcionar almacenamiento local de datos para las aplicaciones individuales y dispositivos, ofreciendo economía, eficiencia, confiabilidad, independencia y simplicidad.
- Trabaja adecuadamente en dispositivos embebidos e IoT.
- Utilizado para analizar grandes cantidades de datos. Posibles usos incluyen análisis de sitios web, análisis estadístico, compilación de métricas de programación y análisis de resultados experimentales.
- Buenos resultados siendo utilizado como almacenamiento de datos en aplicaciones de servidor corriendo en centro de datos.
- Funcionamiento correcto en sitios de tráfico bajo-medio (99,9% de los sitios web actuales).

- Es mono-usuario, es decir, no permite concurrencia de conexiones. No tiene tipos de datos.
- No hay gestión de usuarios. La seguridad se basa en el sistema de permisos de ficheros establecido por Unix/Linux.

MYSQL

- Rápido, sólido y flexible.
- Utilizado por grandes corporaciones como Yahoo, Google, Motorola, entre otras, y en proyectos como Wikipedia, WordPress, Drupal y Joomla.
- Es un gestor de base de datos, y cada tabla está en un fichero diferente, así como vistas y otros objetos.
- Permite múltiples consultas y modificaciones al mismo tiempo.
- Tiene varios tipos de datos según qué tipo de información necesitemos manejar.
- Posibilidad de gestionar usuarios con diferentes niveles de acceso.
- Fácil integración con aplicaciones desarrolladas en C y C++, lenguajes bajo los cuales está desarrollado.
- Sistema cliente-servidor.

PostgreSQL

- Gestor de código abierto
- Control de concurrencia (MVCC) Permite trabajar con grandes volúmenes de datos
- Soporta la mayoría de la sintaxis SQL.
- Posee integridad referencial e interfaces nativas para lenguajes como ODBC, JDBC, C, C++, PHP, PERL, TCL, PYTHON y RUBY.
- Compatible con sistemas operativos UNIX (Linux) y Windows.

- Ahorros considerables de costos de operación.
- Estabilidad y confiabilidad

Microsoft SQL Server

- Fácil administración bajo entorno gráfico.
- Escalabilidad, estabilidad y seguridad.
- Permite trabajar en modo cliente-servidor.
- Licencia comercial.
- Integridad de datos, optimización de consultas, control de concurrencia, backup y recuperación.

Selección de la base de datos

De acuerdo a las características anteriormente descritas sobre cada una de los gestores de bases de datos analizados, se eligió trabajar con MYSQL, principalmente por sus características multiplataforma, ideales para aplicaciones que manejan diferentes tipos de software y hardware, su gran capacidad de almacenamiento y sus bajos costes de mantenimiento para el funcionamiento de una aplicación habitual.

Diseño de la base de datos

Con el fin de almacenar y gestionar la información del sistema gestor de tráfico, se requirió del uso de una base de datos relacional. Para su diseño, se debe definir la estructura conceptual de la base de datos, sin tener en cuenta la tecnología sobre la cual se implementará. Por esta razón, se hizo uso del modelo entidad-relación.

Modelo Entidad – Relación:

Nos permite visualizar en un modelo conceptual las principales características del diseño, a través de entidades e inter-relaciones.

Entidades:

De acuerdo a los requerimientos del proyecto planteado, las entidades seleccionadas son las siguientes:

Usuario: Contiene la información sobre cada uno de los usuarios del sistema, por medio de cada acceso a la plataforma de buses alimentadores.

Bus alimentador: Contiene la información de todos los buses alimentadores que operan en la troncal de origen, junto al estado actual de operación a lo largo del día.

Ruta alimentadora: Contiene la información de las rutas alimentadoras de todo el sistema, especificando su origen y características físicas como el área de acceso de la plataforma.

Despacho: Contiene la información relacionada a los despachos realizados por el operador, sobre cada ruta alimentadora junto a algunos indicadores de calidad asociados al servicio.

Operador: Contiene la información de registro y autenticación de cada operador del sistema y usuario de la aplicación.

Horario del recorrido: Contiene los tiempos de recorrido estimados para cada ruta alimentadora, previamente establecidos, según el día y franja del recorrido.

Origen: Contiene la información de todos los orígenes o troncales que componen el sistema.

Tiempos de asignación: Contiene la información de las solicitudes de bus enviadas por cada uno de los prototipos conectados en cada ruta alimentadora.

Tipo de día: Contiene la información de los tipos de días de acuerdo a la demanda de pasajeros.

Lista de días: Contiene la lista de días clasificados de acuerdo a la demanda de pasajeros.

Atributos:

Cada una de las entidades tienen un conjunto de características asociadas a un elemento en particular. Los atributos correspondientes a cada entidad se detallan a continuación:

- Pertenecientes a la entidad Usuario: Número de acceso, UID de usuario, Fecha de acceso, Hora de acceso, Código de la ruta, Pasajeros en espera, densidad de pasajeros.
- Pertenecientes a la entidad Bus alimentador: Número de bus, Código de ruta, Estado, Capacidad del bus, Número de despacho.
- Pertenecientes a la entidad Ruta alimentadora: Código de la ruta, Nombre de la ruta, Código de origen, Área, Ultimo despacho.
- Pertenecientes a la entidad Despacho: Número de despacho, Código de la ruta, Número del bus, Fecha de despacho, Hora de despacho, Hora de llegada, Delay, Factor de Confiabilidad, Carga de pasajeros, Frecuencia, Hora de llegada esperada, Tope.
- Pertenecientes a la entidad Operador: Documento de operador, Nombres, Apellidos, Perfil, Código de origen, Contraseña, Tipo de horario.
- Pertenecientes a la entidad Horario de recorrido: Código de franja, Código de ruta, Hora de inicio, Hora final, Tiempo estimado, Tipo de día, Franja.
- Pertenecientes a la entidad Origen: Código de origen, Nombre de origen.
- Pertenecientes a la entidad Tiempo de asignación: Número de solicitud, Código de la ruta, Fecha, Hora, Tiempo de espera, Número de bus, Despachado.
- Pertenecientes a la entidad Tipo de día: Código del tipo, Nombre del tipo.
- Pertenecientes a la entidad Lista de Días: Código de día, Fecha, Tipo de día.

Claves candidatas y primarias:

Para cada una de las entidades seleccionadas, se realiza un análisis de los atributos que cumplen las características para hacer parte de las claves candidatas, y posteriormente se selecciona la clave principal, en caso de haber más de una opción:

- Claves candidatas entidad Usuario: Número de acceso **(CP)**

- Claves candidatas entidad Bus alimentador: Número de bus **(CP)**, Número de despacho (clave foránea).
- Claves candidatas entidad Ruta alimentadora: Código de ruta **(CP)**, Nombre de ruta (tipo de dato String), Último despacho (clave foránea).
- Claves candidatas entidad Despacho: Número de despacho **(CP)**
- Claves candidatas entidad Operador: Documento operador **(CP),** Contraseña.
- Claves candidatas entidad Horario de operador: Tipo de horario **(CP)**
- Claves candidatas entidad Horario de recorrido: Código de franja **(CP)**
- Claves candidatas entidad Origen: Código de origen **(CP)**, Nombre de origen (tipo de dato String).
- Claves candidatas entidad Tiempo de asignación: Número de solicitud **(CP)**
- Claves candidatas entidad Tipo de día: Código del tipo **(CP)**
- Claves candidatas entidad Lista de Días: Código de día **(CP)**

Normalización:

Con el fin de evitar anomalías en la construcción de la base de datos a partir de la errónea selección en sus componentes y basándose en las dependencias funcionales, se efectúa el proceso de normalización:

- Primera forma normal: Establece que los dominios de los atributos deben ser valores atómicos (sólo un valor de atributo por tupla)
- Segunda forma normal: Además de estar en primera forma normal, todo atributo no primo (que no se encuentra en ninguna clave) es completamente dependiente de las claves candidatas.
- Tercera forma normal: Además de cumplir la primera y segunda forma normal, todos los atributos no claves dependen de la clave primaria sin ninguna transitividad.

Las entidades que cumplen el proceso de normalización hasta la tercera forma, con sus respectivos atributos y nombres a utilizar en el montaje de la base de datos, son las siguientes:

Cardinalidad:

Para diseñar la base de datos se tuvieron en cuenta las correspondencias de Cardinalidad para las relaciones n-arias presentes entre cada una de las entidades. Para realizar esta clasificación se analizaron varias situaciones dentro del contexto planteado:

- Un USUARIO puede elegir varias RUTAS ALIMENTADORAS y por lo tanto, también puede utilizar varios BUSES ALIMENTADORES.
- Una RUTA ALIMENTADORA contiene varios BUSES ALIMENTADORES.
- Un BUS ALIMENTADOR recibe varios DESPACHOS
- El OPERADOR de demanda gestiona varias RUTAS ALIMENTADORAS dentro de una misma estación o portal, por lo tanto, también despacha varios BUSES ALIMENTADORES
- Una RUTA alimentadora contiene varios HORARIOS DE RECORRIDO, DESPACHOS y TIEMPOS DE ASIGNACIÓN
- Un ORIGEN puede contener varias RUTAS ALIMENTADORAS y por lo tanto también varios BUSES ALIMENTADORES.

Tablas de la base de datos:

De acuerdo modelo propuesto para la base de datos, el diseño de las tablas, incluyendo los nombres reales utilizados, se muestra a continuación:

Tabla 19. Tabla Usuario

Usuario						
NumAcceso	UID	FechaAcceso	HoraAcceso	CodRuta	PasajerosEspera	Densidad Pasajeros

Tabla 20. Tabla Bus Alimentador

BusAlimentador				
NúmeroBus	CodRuta	Estado	CapacidadBus	NumDespacho

Tabla 21. Tabla Ruta Alimentadora

RutaAlimentadora				
CodRuta	NombreRuta	CodOrigen	Area	UltimoDespacho

Tabla 22. Tabla Despacho

Despacho					
NumDespacho	CodRuta	NumeroBus	FechaDespacho	HoraDespacho	HoraLlegada
Delay	FactorConfiabilidad	CargaPasajeros	Frecuencia	HoraLlegadaEsperada	Tope

Tabla 23. Tabla Operador

Operador						
DocOperador	Nombres	Apellidos	Perfil	CodOrigen	Contraseña	TipoHorario

Tabla 24. Tabla Horario Recorrido

HorarioRecorrido						
CodFranja	CodRuta	HoraInicio	HoraFin	TiempoEstimado	TipoDia	Franja

Tabla 25. Tabla Origen

Origen	
CodOrigen	NombreOrigen

Tabla 26. Tabla Tiempos de Asignación

TiempoAsignacion						
NumSolicitud	CodRuta	Fecha	Hora	TiempoEspera	NumeroBus	Despachado

Tabla 27. Tabla Tipo de Día

TipoDia	
TipoDia	NombreTipoDia

Tabla 28. Tabla Lista de Días

ListaDias		
CodDia	Fecha	TipoDia

Diagrama del modelo:

En la siguiente figura se observa el diseño propuesto para el modelo entidad-relación, indicando los tipos de datos para cada uno de los atributos que componen la tabla de la base de datos, y la relación entre cada entidad planteada:

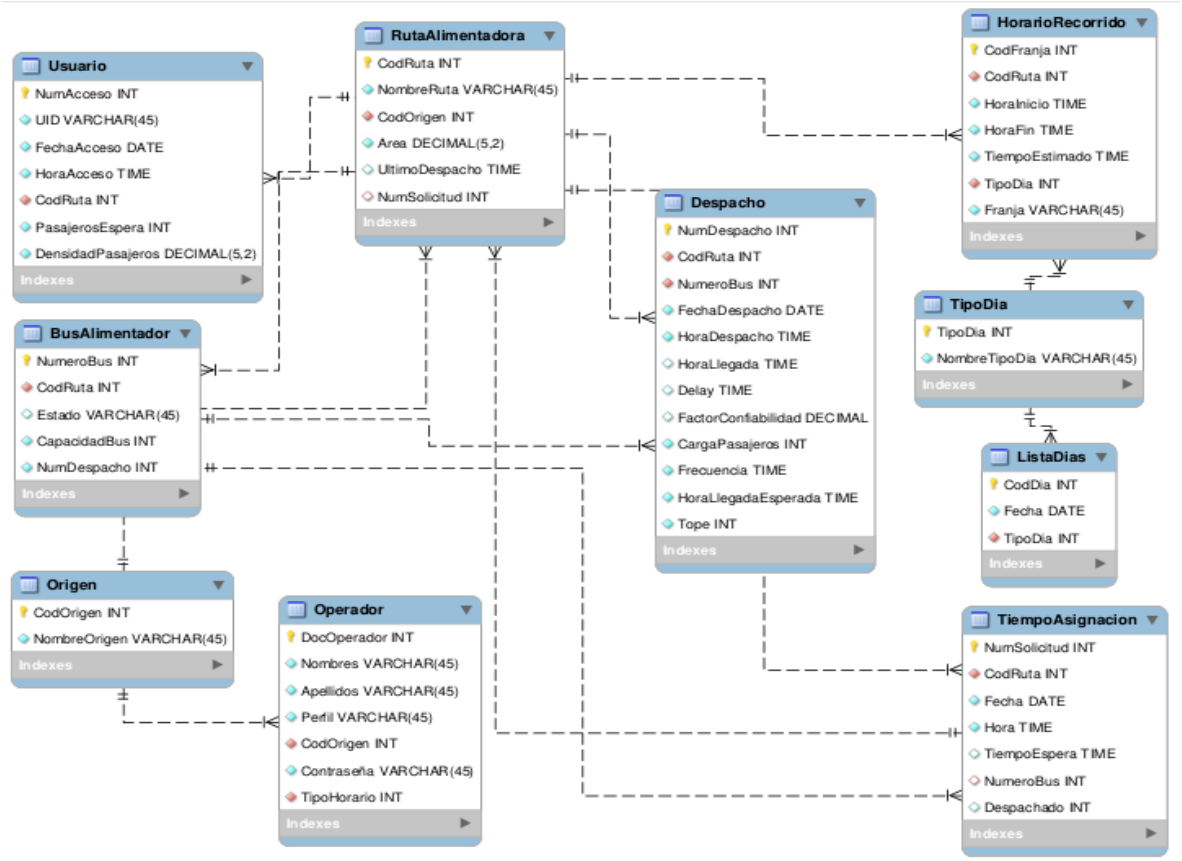

Figura 53. Modelo Entidad-Relación

Desarrollo de la aplicación de usuario

El desarrollo de la aplicación web se encuentra conformado por varias fases: desde la definición de los requisitos de la aplicación donde se analiza el problema y se describe la función que debe cumplir la aplicación, la fase de diseño que contiene toda la información relacionada con la estructura y contenido de la aplicación, la fase de pruebas que describe el conjunto de

validaciones y verificaciones realizadas a la aplicación y por último, la fase de despliegue de la aplicación en un entorno real.

Requisitos

El objetivo de la aplicación web es ofrecer una herramienta que permita visualizar de manera organizada, clara y concisa la información relacionada a la gestión de buses alimentadores. Esta aplicación permite obtener información de varias maneras, de acuerdo a la necesidad o al rol del usuario que accede. Para cumplir con la finalidad planteada, la aplicación debe cumplir con las siguientes condiciones:

- Ofrecer datos en tiempo real que le permiten al operador de rutas gestionar la demanda de las distintas rutas alimentadoras mediante la recepción de las solicitudes enviadas por los módulos, el despacho de buses disponibles, así como el registro de la entrada y salida de cada bus alimentador que realiza un recorrido especifico.
- Datos históricos organizados y clasificados para ser visualizados de forma gráfica, con el fin de evaluar a partir de ellos los posibles cambios a las frecuencias de los buses alimentadores, buscando mejorar los indicadores de calidad del sistema.
- Interfaz de administración que le permite al administrador o usuarios autorizados acceder a la información más relevante y sensible del sistema, con el fin de agregar, editar o eliminar contenido de la aplicación.
- Garantizar un nivel de seguridad adecuado respecto al acceso, manipulación y confidencialidad de la información contenida en la aplicación. Para eso, los usuarios deben proveer credenciales válidas de autenticación a la aplicación para acceder a las áreas más sensibles y críticas en la operación del sistema.

- Un entorno visual agradable y claro para el usuario, que le permita tomar acciones rápidas de acuerdo a la información recibida a través de la página.

Diseño

Diagrama de flujo:

Para iniciar la codificación de la aplicación, se desarrolló un diagrama de flujo con el fin de representar la lógica y las actividades que debe contener. Además, esto nos permitió dividir la complejidad del proceso en etapas más pequeñas, logrando una mejor interpretación del problema. El diagrama está compuesto por cuatro bloques principales: Llegada de pasajero, solicitud de bus, despacho de bus y llegada de bus.

Llegada de pasajero:

Este diagrama representa el flujo de información a partir de la llegada del pasajero a la plataforma y de su registro en el módulo de gestión:

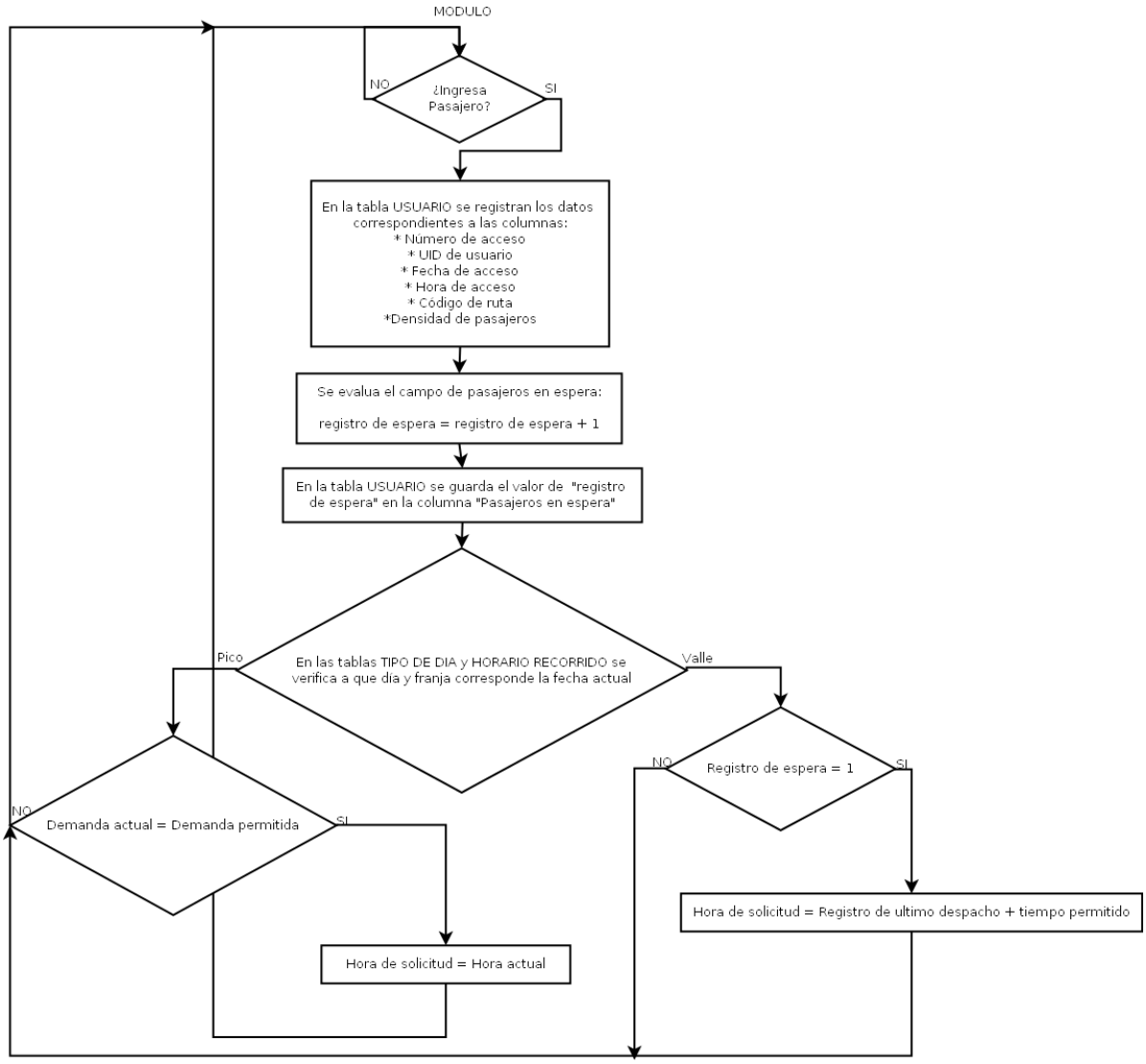

Figura 54. Diagrama de flujo - Llegada de pasajero

Solicitud de bus:

Este diagrama representa la información que envía el módulo de gestión NFC al servidor central, a partir de los datos recolectados del ingreso de pasajeros:

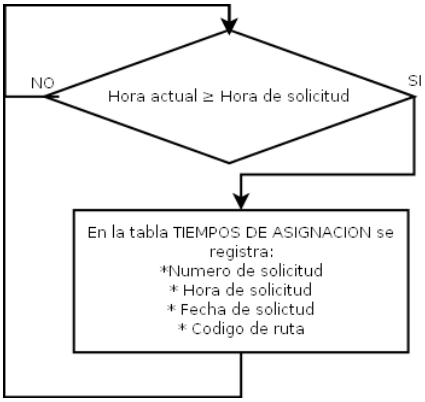

Figura 55. Diagrama de flujo - Solicitud de bus

Despacho de bus:

Representa la información que se registra en la base de datos a partir del despacho de un bus por parte del operador de rutas.

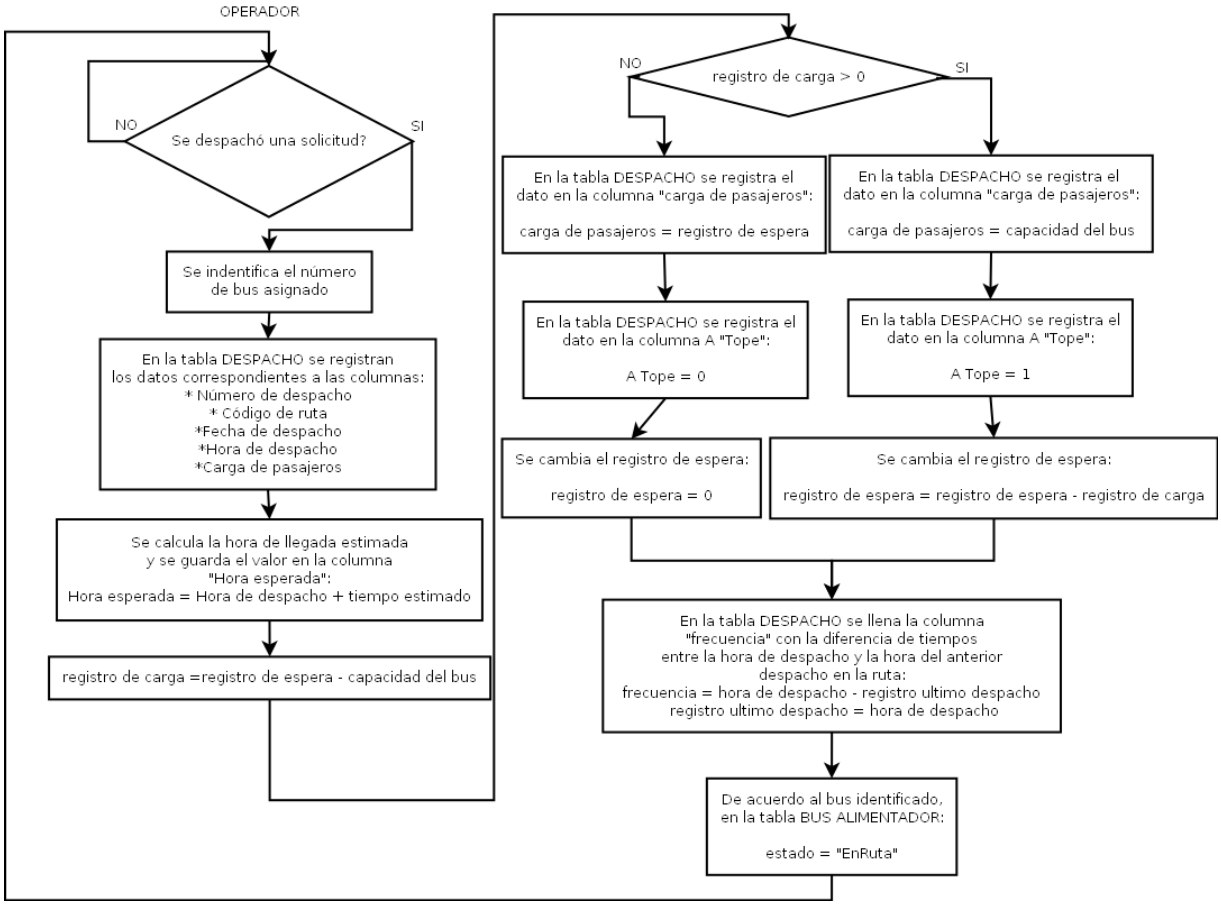

Figura 56. Diagrama de flujo - Despacho de bus

Llegada de bus:

Relaciona la información referente a la llegada de un bus alimentador al realizar ruta.

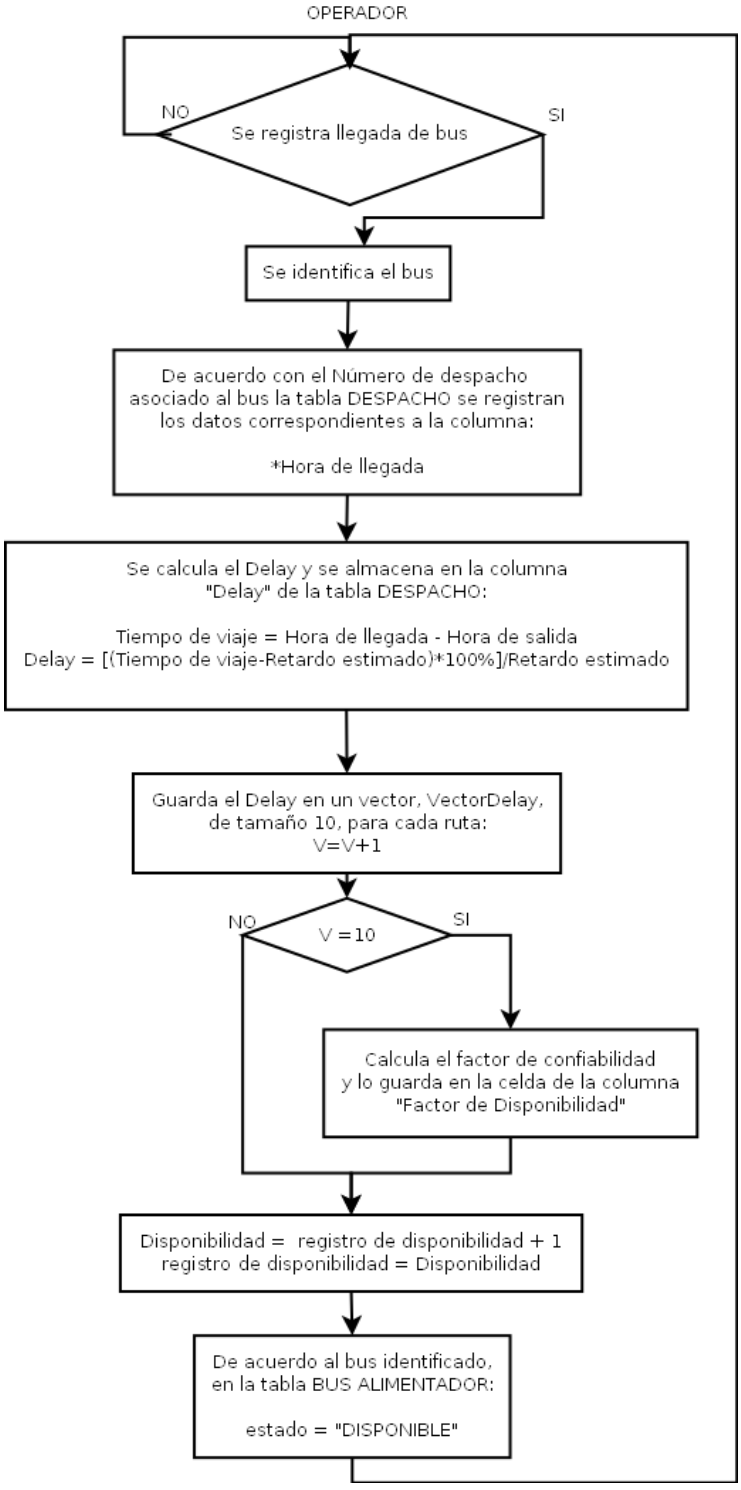

Figura 57. Diagrama de bus - Llegada de bus

Para la aplicación de usuario se buscó elegir herramientas de desarrollo ágiles, potentes, y compatibles con el resto de las aplicaciones utilizadas a lo largo del proyecto. Por esta razón, vimos la necesidad de trabajar con framework web, pues garantizan un desarrollo bajo buenas prácticas, acelerando el proceso desarrollo y ofreciendo elementos de fácil reconfiguración y adaptables a otros entornos de desarrollo.

Se eligieron los framework Django y Flask como framework web de desarrollo. En el caso de Django, se utilizó para desarrollar la aplicación pues ofrece una serie de características de gran potencial para desarrollar una aplicación robusta:

- Proyecto de código abierto.
- Patrón de arquitectura de software MVC
- Flexibilidad
- Reutilización de código (plantillas heredadas)
- Acoplamiento débil
- Completamente programado en lenguaje Python
- Abstracción del motor de base de datos y su adaptador de conexión

Por el lado de Flask, es un microframework que permite crear aplicaciones web rápidamente y con un número mínimo de líneas de código, ya que incluye un servidor web de desarrollo para poder corroborar el funcionamiento de la aplicación sin tener que instalar un servidor adicional. Flask cuenta con las siguientes características, tenidas en cuenta para escoger este microframework para el desarrollo [1]:

- Propio servidor web
- Debbuging
- Fácil desarrollo
- Gran cantidad de documentación

- Compatibilidad con PyGal
- Compatibilidad con WSGI (**Web Server Gateway Interface**) es una interfaz entre servidores web y aplicaciones web o frameworks para el lenguaje de programación Python.

Gestión de datos en tiempo real

Instalación y configuración de Django:

Previamente a la instalación de Django, se requirió de un conjunto previo de herramientas instaladas, entre ellas Python, el motor de base de datos MYSQL y el conector MySQLdb. Posteriormente, se realizó la instalación de la versión oficial 1.10:

sudo pip install Django==1.10

Luego, se creó un directorio en el servidor, donde se alojaron los proyectos necesarios para el desarrollo de la aplicación.

cd /home/johngutierrez9891/djcode

Desde este directorio, se ejecutaron el comando para crear el proyecto:

django-admin.py startproject nombre_del_proyecto

Una vez creado el proyecto, se creó automáticamente el directorio del proyecto que contiene los archivos para interactuar con el proyecto, importar archivos externos, configuraciones principales, declaración de las URL y la información de compatibilidad con el servidor web. Es recomendable instalar el directorio previo en un directorio fuera de la carpeta raíz. Esto para evitar que desde la web se puede acceder a esta información.

Django ofrece un servidor de desarrollo permitiendo llevar a cabo la creación de la aplicación sin necesidad de configurar un servidor de producción. Para ejecutar se ingresa el comando:

python manage.py runserver

Es posible pasar como argumentos a este comando la dirección IP que escucha el servidor y el puerto del servidor:

python manage.py runserver 0.0.0.0:8080

En este momento, se verifica accediendo desde el navegador web a la dirección IP, que el servidor se encuentre funcionando correctamente.

Conexión de Django con la base de datos:

Dentro del patrón MTV que maneja Django, el modelo hace referencia al acceso a la capa de datos, por lo cual la configuración de la capa de base de datos requiere especial atención. Para indicarle a Django cómo efectuar la conexión con la base de datos se abrió el archivo settings.py y se editaron los campos de configuración de la base de datos, de acuerdo al motor de base de datos utilizado, el nombre de la base de datos, el usuario administrador que va a manipular la base de datos, la contraseña de acceso a la base de datos, el host a usar para conectarse a la base de datos y el puerto para realizar la conexión.

Si la base de datos está sobre la misma computadora que la instalación de Django se deja vacía, mientras que si se deja el puerto vació, el adaptador de base de datos (MySQLdb) usará el puerto por omisión acorde al servidor de base de datos:

```
DATABASES = {
   'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'DJANGO',
        'USER': 'root',
        'PASSWORD': 'xxxxxxx',
        'HOST':'',
        'PORT': '',
   }
}
```

Para probar la configuración de la base de datos, se puede validar la conexión ingresando al shell de python desde el directorio del proyecto de Django:

```
python manage.py shell
from django.db import connection
cursor = connection.cursor()
```

Creación de la aplicación:

Siendo la aplicación, el conjunto de archivos de código fuente que contiene las vistas y modelos que contendrán la información del proyecto, se procede a acceder al directorio creado inicialmente y se ingresa el comando para crear la aplicación:

```
python manage.py startapp nombre_de_la_aplicación
```

El comando anterior crear un directorio que contiene los archivos necesarios para alojar los modelos y vistas de la aplicación. En el archivo models.py se ingresa la información de los modelos que, a través de las clases predeterminadas de Django, interactúan con la base de datos para la creación y modificación de las tablas.

Para activar los modelos en el proyecto Django, se agregó la aplicación a la lista de aplicaciones instaladas en el archivo de configuración. Se editó la variable INSTALLED_APPS del archivo settings.py:

```
INSTALLED_APPS = [
   'django.contrib.admin',
   'django.contrib.auth',
   'django.contrib.contenttypes',
   'django.contrib.sessions',
   'django.contrib.messages',
   'django.contrib.staticfiles',
   'nombre_de_la_aplicacion',
]
```

Luego de la activación de la aplicación en el archivo de configuración, se generó el archivo de migración correspondiente y se aplicaron los cambios a la base de datos:

```
python manage.py migrations nombre_de_la_aplicacion
python manage.py migrate nombre_de_la_aplicacion
```

Para verificar las sentencias SQL generadas por la migración anterior se verificó con:

python manage.py sqlmigrate nombre_de_la_aplicacion nombre_de_la_migracion

El archivo models.py contiene toda la información relacionada con el diseño de la base de datos.

Interfaz de administración:

Django provee una atractiva interfaz de administración que requiere ser configurada adicionalmente. Para esto, se hace uso del archivo admin.py que contiene la información de los modelos que son visualizados en la interfaz de administración. Cada modelo tiene asociada una clase diferente con los atributos seleccionados para cada uno.

Después de configurar las clases, se realiza la migración para instalar todas las tablas de la base de datos que la interfaz de administración requiere. Adicional, se creó un usuario administrador:

python manage.py createsuperuser

Para hacer modificables las tablas de nuestra aplicación desde la interfaz de administración, se configuraron las interfaces de los objetos de Django que representan cada modelo modificando el archivo admin.py:

admin.site.register(Origen, OrigenAdmin)
admin.site.register(RutaAlimentadora, RutaAlimentadoraAdmin)
admin.site.register(TipoDia,TipoDiaAdmin)
admin.site.register(ListaDias,ListaDiasAdmin)
admin.site.register(HorarioRecorrido,HorarioRecorridoAdmin)
admin.site.register(Usuario,UsuarioAdmin)
admin.site.register(TiempoAsignacion,TiempoAsignacionAdmin)
admin.site.register(BusAlimentador, BusAlimentadorAdmin)
admin.site.register(Despacho, DespachoAdmin)
admin.site.register(Operador)

Plantillas de la aplicación:

Las siguientes plantillas fueron utilizadas en el desarrollo de la aplicación web:

- **Página de acceso y autenticación:**

Esta es la plantilla raíz y le permite al usuario realizar la autenticación en la aplicación. Si el usuario no se encuentra autenticado todavía, se muestra un mensaje de advertencia y se habilita un enlace a la página de autenticación, donde debe ingresar su usuario y contraseña. No existe ninguna opción para registrar nuevos usuarios desde esta ventana, ya que estas funciones son asignadas al administrador del sistema:

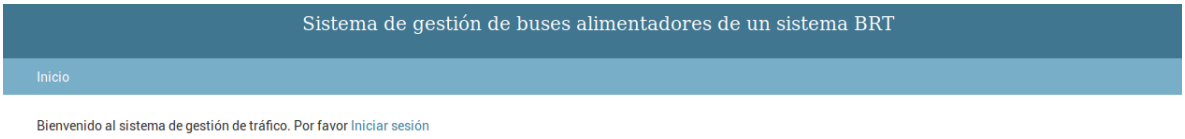

Figura 58. Plantilla de inicio

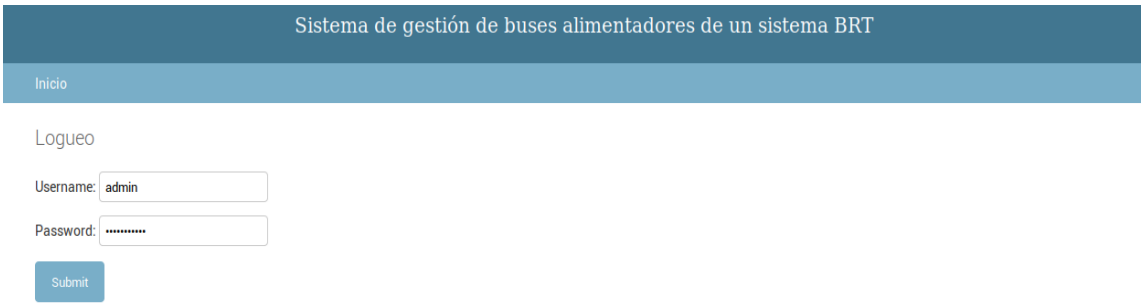

Figura 59. Plantilla de autenticación

- **Página home:**

Esta plantilla es la página central de la aplicación, donde se encuentra la información general de la aplicación y los enlaces a los diferentes módulos. En la parte superior aparece el usuario con el cual se está accediendo, así como la opción para salir del sistema. En la parte central (recuadro azul), se describe brevemente el propósito de la aplicación. Abajo de esta, aparecen dos secciones

adicionales que especifican el rol de cada uno de los módulos internos. En la parte derecha de la plantilla se encuentra el menú de enlaces para ingresar a cada una de los módulos de la aplicación, así como un enlace al video de simulación realizado el VISSIM.

Finalmente, en la parte inferior aparece la información realizada a los creadores de la página y enlaces de interés a las páginas institucionales de la universidad:

Figura 60. Plantilla de página principal

- **Sistema de gestión en tiempo real**

Esta plantilla contiene la información en tiempo real de las solicitudes de buses recibidas desde los módulos NFC dispuestos en cada ruta alimentadora, así como los buses en tránsito. El operador de las rutas interactúa con esta información despachando los buses a las rutas requeridas y registrando la llegada de los buses a la estación:

Figura 61. Plantilla del sistema de gestión en tiempo real

- **Despacho de rutas:**

Esta plantilla está diseñada para permitirle al operador despachar una ruta alimentadora, luego de seleccionar una de las solicitudes enviadas por los módulos NFC. Aquí, puede seleccionar uno de los buses disponibles de la lista desplegable, y luego seleccionar el botón "Despachar bus" para enviar la información:

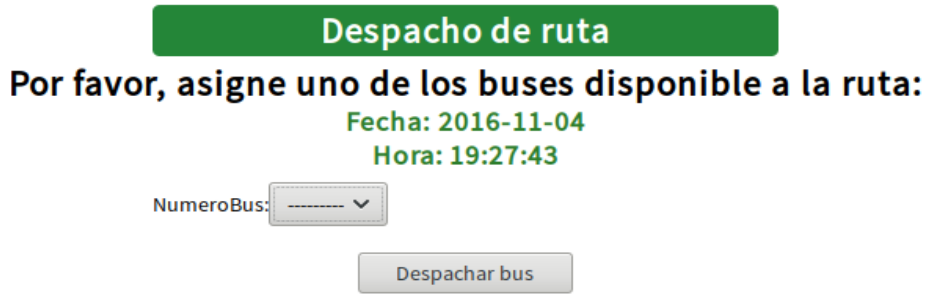

Figura 62. Plantilla de despacho de rutas

- **Llegada de buses:**

Esta plantilla está diseñada para que el operador pueda registrar los buses que finalizar el recorrido asignado y llegan a la estación para volver a estar disponibles para un nuevo recorrido:

Figura 63. Plantilla de llegada de buses

- **Acceso a la página de administración**

Esta plantilla permite la autenticación al administrador del sistema, quien es el encargado de administrador la información contenida en la base de datos:

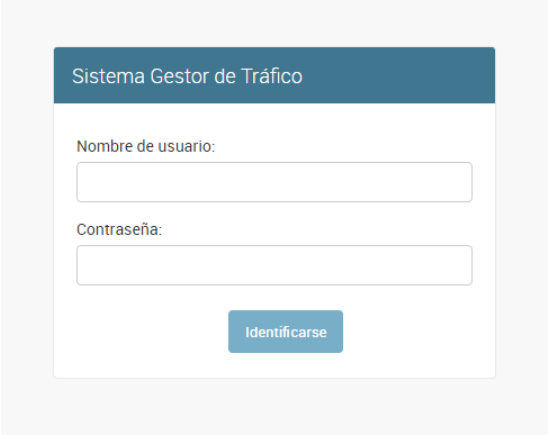

Figura 64. Plantilla de autenticación a la página de administración

- **Página de administración:**

En la siguiente figura, se observa la interfaz de administración utilizada para gestionar la información de la aplicación contenida en la base de datos:

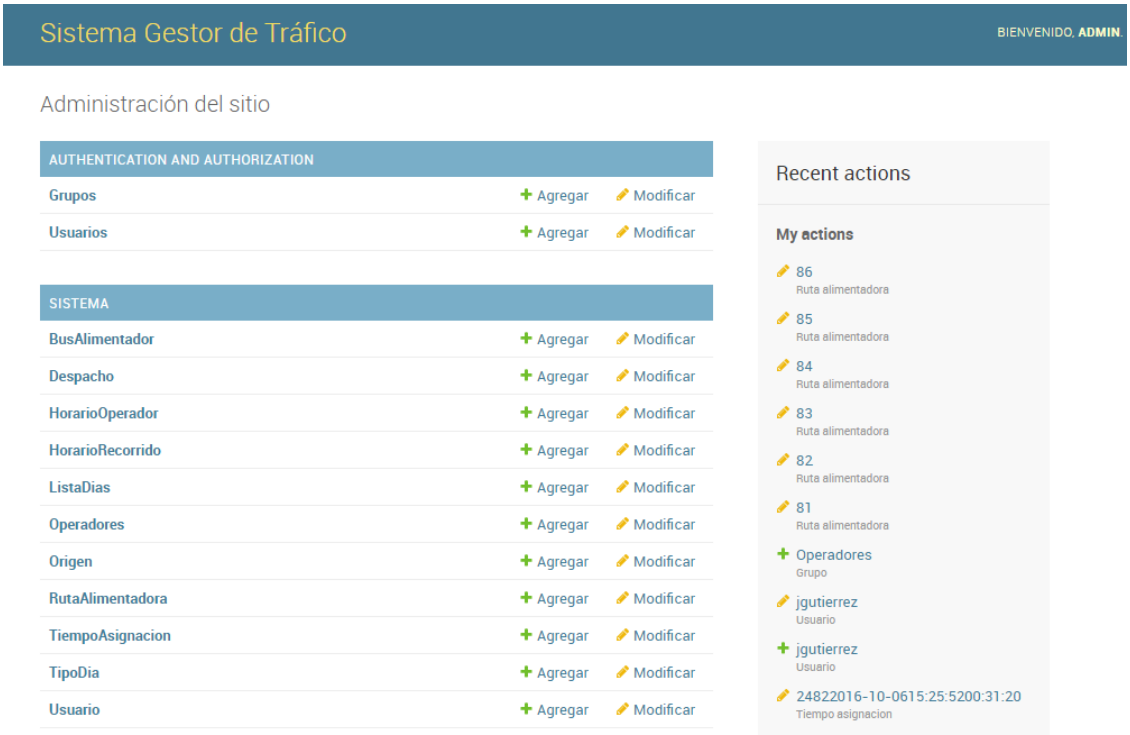

Figura 65. Plantilla de la interfaz de administración

Desarrollo de la aplicación para la visualización de datos históricos

Esta etapa de la aplicación fue desarrollada en Flask. El principal formato abierto para gráficos vectoriales es SVG (Scalable Vector Graphics) o Gráficos Vectoriales Redimensionables. PyGal es una librería de código abierto de Python que permite dibujar graficas SVG mediante comandos de Python. SVG es el formato de grafico usado en esta aplicación web para visualizar los datos. La aplicación web debe consultar los datos para crear las tuplas que serán graficadas y posteriormente exportarlas en un formato SVG para insertarlas en una plantilla html.

Instalación y configuración

Para evitar problemas de incompatibilidad de versiones de librerías se debe crear un entorno virtual de Python, mediante la herramienta VirtualEnv. Antes de crear un entorno virtual se creó una carpeta para alojar el entorno, llamada "consultor". Para instalar VirtualEnv, desde el emulador de terminal, se ingresó la siguiente línea desde la carpeta anteriormente mencionada:

sudo apt-get install python-virtualenv

Luego, se creó el entorno virtual con:

virtualenv flask

Para crear la estructura básica de la aplicación web se crearon los siguientes directorios, dentro de la carpeta "consultor":

mkdir app
mkdir app/static
mkdir app/templates
mkdir tmp

La carpeta "static" contiene las imágenes y demás archivos insertados en las plantillas HTML. La carpeta "templates" contiene las plantillas HTML y archivos de estilo CSS. Luego, creamos el siguiente script Init dentro de la carpeta app, llamado __init__.py:

from flask import Flask

```
app = Flask(__name__)
from app import views
```

Este script se encarga de crear el objeto de la aplicación e importar el módulo "views". Este módulo se encargará de la gestión de la aplicación web mediante comandos Python. Para crear el modulo se debe crear un archivo formato .py. En el módulo "views" se crean funciones que de acuerdo al diseño estarán mapeadas hacia url's. En principio el módulo "views" contiene el siguiente texto:

```
from app import app
@app.route('/')
@app.route('/index')
def index():
        return "Hello, World!"
```

En las líneas de texto anteriores se crea la función Flask, la cual genera una ruta URL y retorna un texto. Más adelante estas funciones retornaran y administraran plantillas HTML. Finalmente, para que la aplicación web funcione se creó un script para inicializar el servidor web de desarrollo. En nuestro caso, se llamó run.py, y se encuentra dentro de la carpeta que contiene el entorno virtual. Run.py contiene el siguiente texto inicialmente:

```
#!flask/bin/python
from app import app
app.run(debug=True)
```

Después en el script _init_.py se importaron las librerías utilizadas en la aplicación web y el script contiene las siguientes líneas al comienzo:

```
# -*- coding: utf-8 -*-
from flask import Flask, render_template, render_template_string, redirect,url_for, request, flash, session
from datetime import datetime, timedelta, time
import time
from time import gmtime, strftime
from wtforms import Form
from passlib.hash import sha256_crypt
```

```
from MySQLdb import escape_string as thwart
from dbconnect import connection
import string
import pygal
from app import app
```

Como se observa dentro de los módulos importados aparece uno llamado “dbconect”. Este módulo se crea en la carpeta app y se usa para crear la conexión con la base de datos. El modulo dbconect.py contiene la siguiente función:

```
import MySQLdb
def connection():
        conn = MySQLdb.connect(host="localhost",user = "root", passwd = "servidorgestor", db = "PROYECTO")
        c = conn.cursor()
        return c, conn
```

Flask trabaja con formularios y plantillas. Para habilitar el uso de formularios se debe crear un módulo, en este caso se llamó “config.py”, que contiene solamente dos líneas:

```
WTF_CSRF_ENABLED = True
SECRET_KEY = 'you-will-never-guess'
```

Estas líneas aumentan la seguridad en el envío de formularios. El texto “you-will-never-guess” se debe cambiar por una contraseña secreta. Una herramienta útil de Flask son los mensajes Flash, los cuales permiten enviar datos desde las funciones Flask hacia las plantillas HTML, con el objetivo de ser visualizados como textos.

El diseño de la aplicación de visualización histórica consta de 13 plantillas. El usuario debe autenticarse y elegir las opciones correspondientes para acceder a cada una, mas no podrá hacerlo directamente. El siguiente diagrama de flujos muestra la ubicación de cada plantilla:

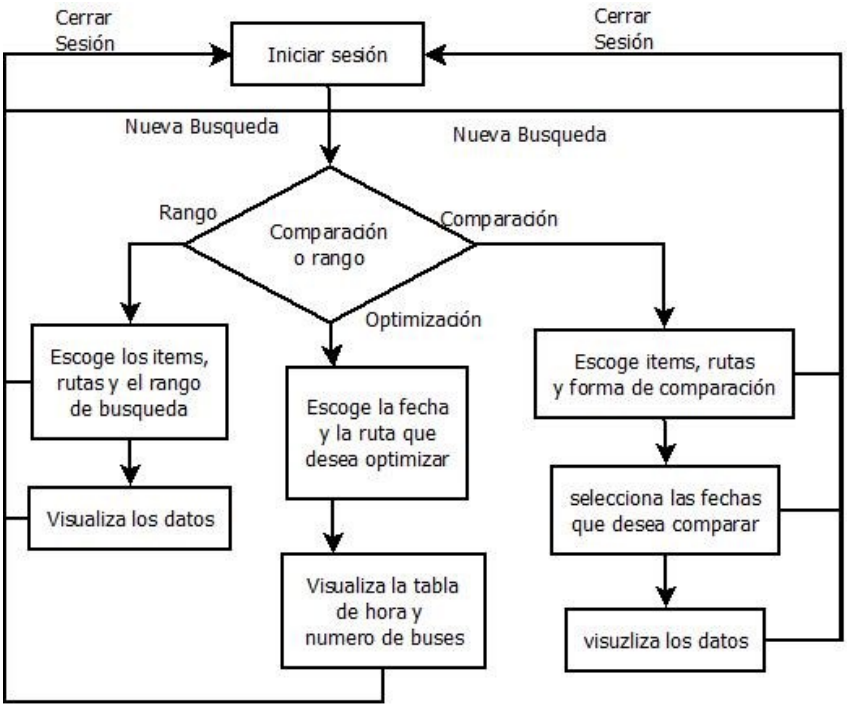

Figura 66. Diagrama de flujo de acceso a las plantillas del histórico de datos.

Plantillas para la visualización histórica de datos

A continuación, se describen las funciones y características de cada plantilla:

- **Iniciar sesión:**

En esta plantilla el usuario deberá autenticarse con su documento de identidad y una contraseña asociada. Si no se autentica no podrá acceder a ninguna plantilla.

Figura 67. Plantilla de autenticación

- **Comparación, rango u optimización**

En esta plantilla el usuario deberá elegir una forma de visualizar los datos. Puede escoger visualizar un rango de fechas y horas, visualizar diferentes fechas en una gráfica para compararlas o evaluar el comportamiento de un día y una ruta para optimizar su asignación de recursos. Tiene la posibilidad de cerrar sesión desde esta plantilla:

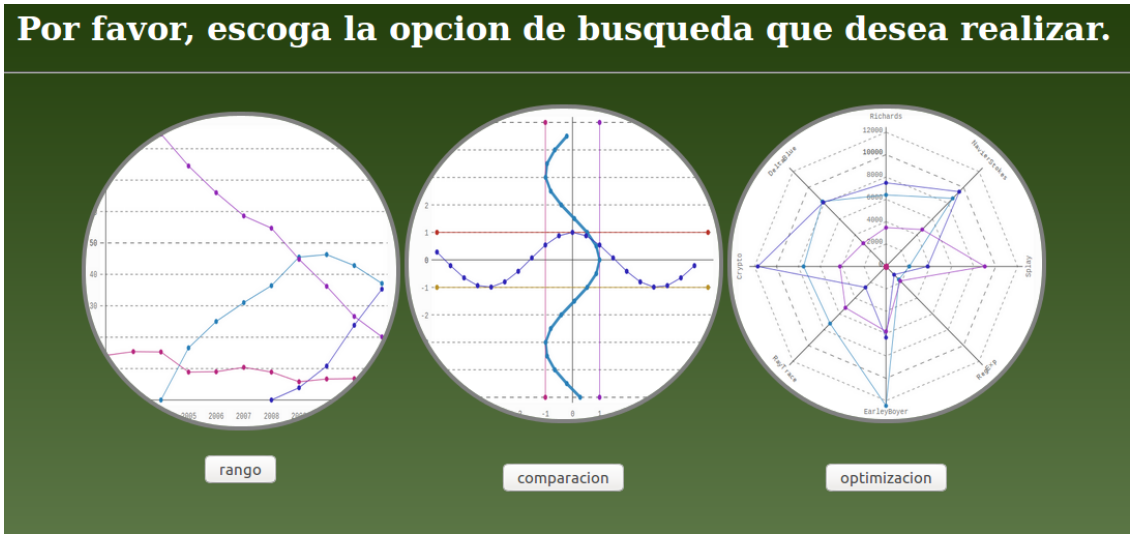

Figura 68. Plantilla de tipo de búsqueda

- **Selección de ítems, rutas y rango de búsqueda**

Esta plantilla se despliega si el usuario selecciona la opción "rango" en la plantilla anterior. Aquí, se deberá seleccionar al menos un KPI, una ruta y el rango de fechas y horas que desea visualizar. El usuario tiene la posibilidad de cerrar sesión desde esta plantilla, así como realizar una nueva búsqueda:

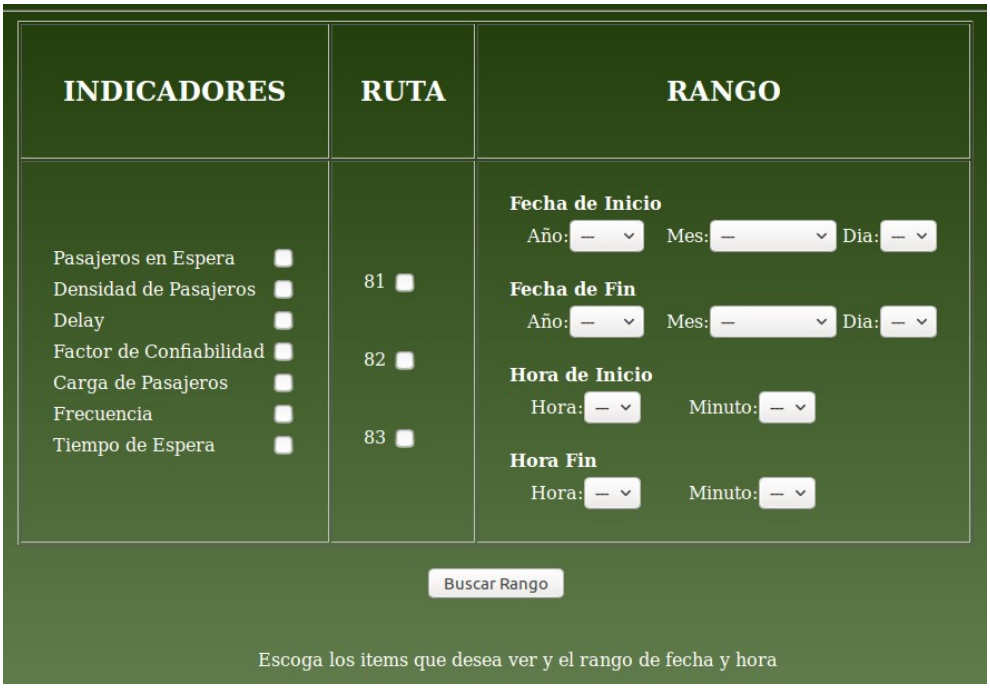

Figura 69. Plantilla de selección de ítems, rutas y rango de búsqueda

- **Visualización de datos (rango)**

En esta plantilla se visualizará una gráfica por cada indicador, y en cada grafica una curva por cada ruta seleccionada. El dominio de cada grafica corresponde al tiempo entre fechas y horas seleccionadas. Desde esta plantilla es posible cerrar sesión o hacer a una nueva búsqueda:

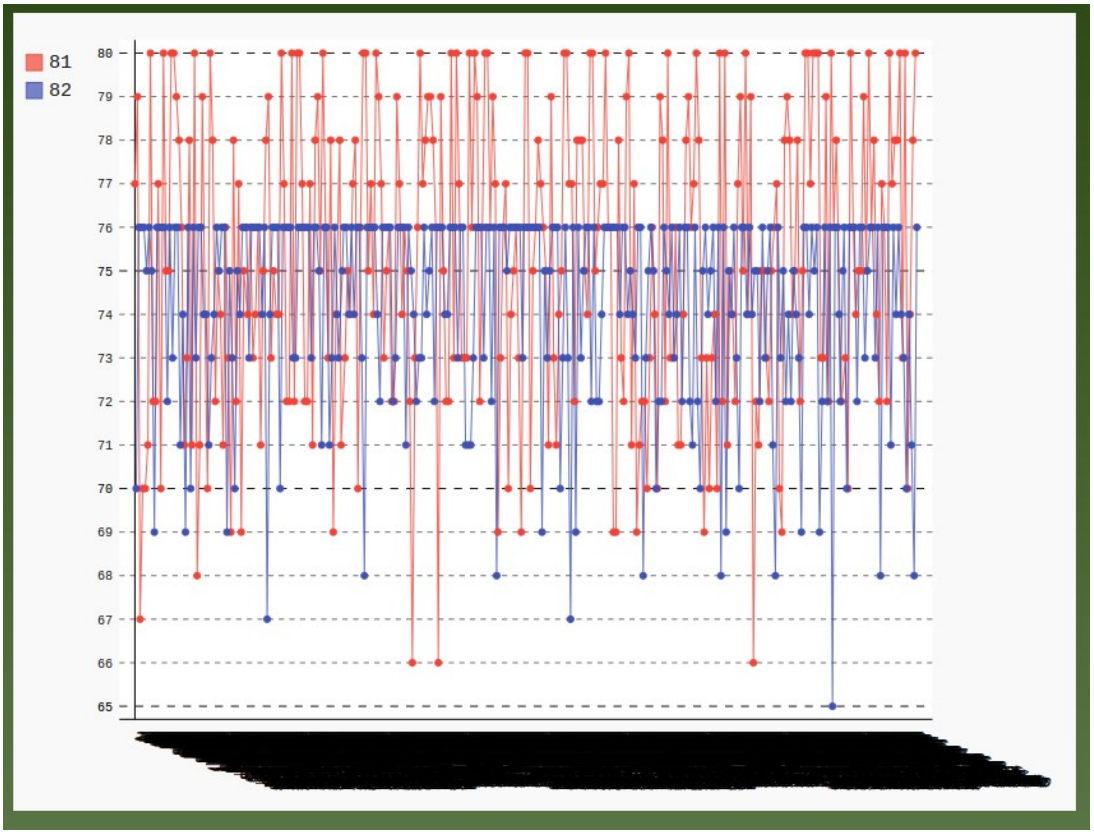

Figura 70. Plantilla gráfica de un indicador y dos rutas para un rango a priori

- **Selección de ítems, rutas y tipo de comparación:**

Si el usuario escoge visualizar los datos en forma de comparación, en esta plantilla deberá escoger al menos un KPI, una ruta y el tipo de comparación de los datos. Desde esta plantilla es posible cerrar sesión o hacer a una nueva búsqueda:

Figura 71. Plantilla de selección de ítems, rutas y tipo de comparación

- **Selección de fechas a comparar:**

Si el usuario desea comparar algunos días aparecerá la siguiente planilla para que seleccione los días. Deberá hacer clic en cada checkbox ubicado al frente al día que desea comparar:

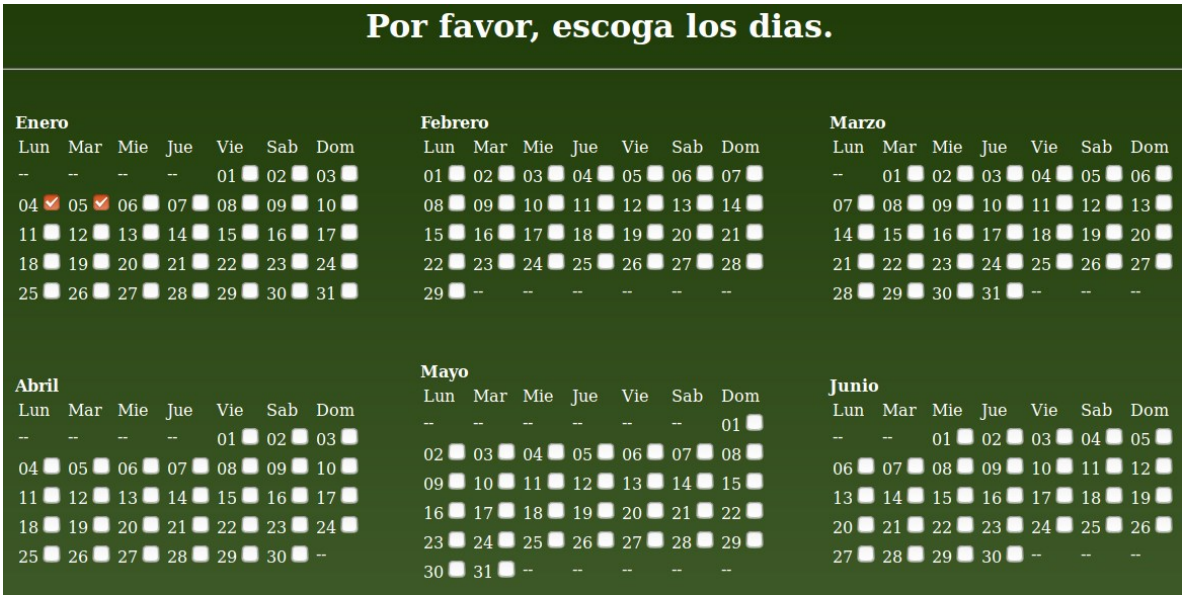

Figura 72. Plantilla de selección de fechas a comparar

- **Selección de tipos de día a comparar:**

Si el usuario desea comparar los días pertenecientes a un tipo específico, aparecerá la siguiente plantilla para realizar la selección:

Figura 73. Plantilla de selección de tipos de día a comparar

- **Selección de meses a comparar:**

Si el usuario desea comparar dos o más meses, aparecerá la siguiente plantilla para realizar la selección. Deberá hacer clic en cada checkbox que aparece debajo del mes a comparar:

Figura 74. Plantilla de selección de meses a comparar

- **Selección de semanas a comparar:**

Si el usuario desea comparar dos o más semanas, aparecerá la siguiente plantilla para realizar la selección. Deberá hacer clic en cada checkbox al frente de la semana que desea comparar:

Figura 75. Plantilla de selección de semanas a comparar

- **Visualización de datos (Comparación):**

En esta plantilla se visualizará una gráfica por cada indicador y por cada ruta, y en cada grafica se mostrará una curva por cada fecha seleccionada. El dominio de cada grafica son las horas de cada día:

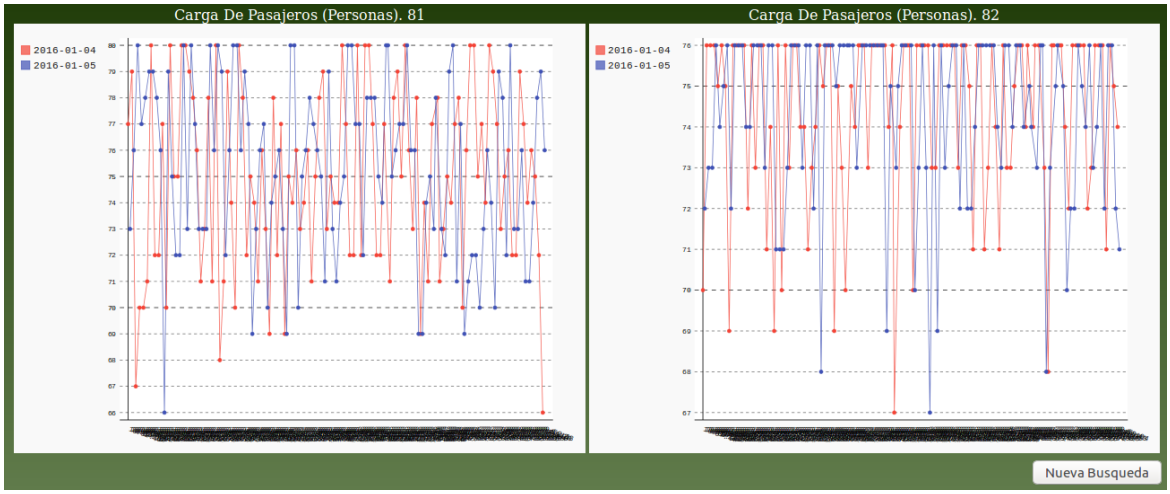

Figura 76. Plantilla de dos gráficas, seleccionando un indicador, dos rutas y dos fechas

- **Selección de fecha y ruta para optimizar:**

Si el usuario escoge la opción, en esta plantilla deberá indicar la fecha y la ruta que desea optimizar. Desde esta plantilla es posible cerrar sesión o hacer a una nueva búsqueda:

Figura 77. Plantilla de selección de fecha y ruta a optimizar

- **Resultado de la optimización:**

En esta plantilla se visualizará una tabla que indicará el número de buses que deben operar a cierta hora del día:

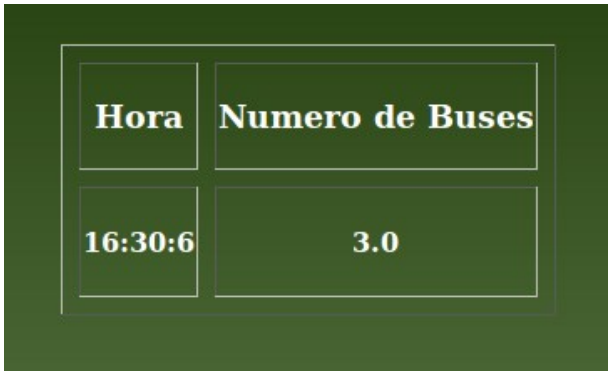

Figura 78. Plantilla del resultado de la optimización

Algoritmo de optimización:

El objetivo del algoritmo de optimización es analizar las curvas de demanda de las rutas alimentadoras para identificar las horas pico y las horas valle, y proponer un número de buses

necesarios para satisfacer la demanda en estas diferentes franjas horarias, con el fin de evacuar los pasajeros a sus destinos.

Como no existen datos de la demanda de pasajeros, primero se diseñó un algoritmo que simulara la llegada de pasajeros. Este algoritmo entrega un vector que proporciona el momento exacto en el que llega un pasajero. El tiempo es trabajado en segundos, de este modo las 05:00 am corresponde a 18000 segundos y las 09:30 pm a 77400 segundos.

Partimos de las franjas pico y valle propuestas por Transmilenio. Como se muestra en la siguiente figura, existen 4 franjas horarias categorizadas como valle y tres franjas horarias categorizadas como pico. Esta categorización se debe a la demanda de usuarios prevista por Transmilenio, donde es mayor en las franjas horarias pico y menor en las franjas horarias valle:

Figura 79. Horario de operación del sistema Transmilenio

Al convertir las horas de las franjas horarias en segundos se obtuvo la siguiente tabla:

Tabla 29. Conversión de franjas horarias a segundos

Hora	Segundos
6:00 am	21600
8:30 am	30600
9:30 am	34200
3:30 pm	55800
4:30 pm	59400
7:30 pm	70200

El algoritmo para la creación de una señal que muestre el comportamiento de la demanda de pasajeros funciona bajo las siguientes reglas:

- En las horas valle se registra la llegada de los pasajeros en intervalos de tiempo aleatorios entre 30 y 60 segundos, y en las horas pico se registra la llegada de los pasajeros en intervalos de tiempo aleatorios entre 1 y 4 segundos. Se escogió el tiempo aleatorio de 1 a 4 segundos ya que, según los datos entregados por Transmilenio, en una franja horaria pico se tiene una demanda de pasajeros de, aproximadamente, un pasajero cada 2,6 segundos.
- El tiempo aleatorio para la hora valle se escogió a priori, observando el comportamiento típico. El algoritmo tiene en cuenta los intervalos de tiempo en los cuales cambia la franja horaria valle a pico y viceversa, y para estos intervalos se registra la llegada de un pasajero en un tiempo aleatorio de 3 a 6 segundos.

El comportamiento observado con este algoritmo se muestra en la siguiente figura:

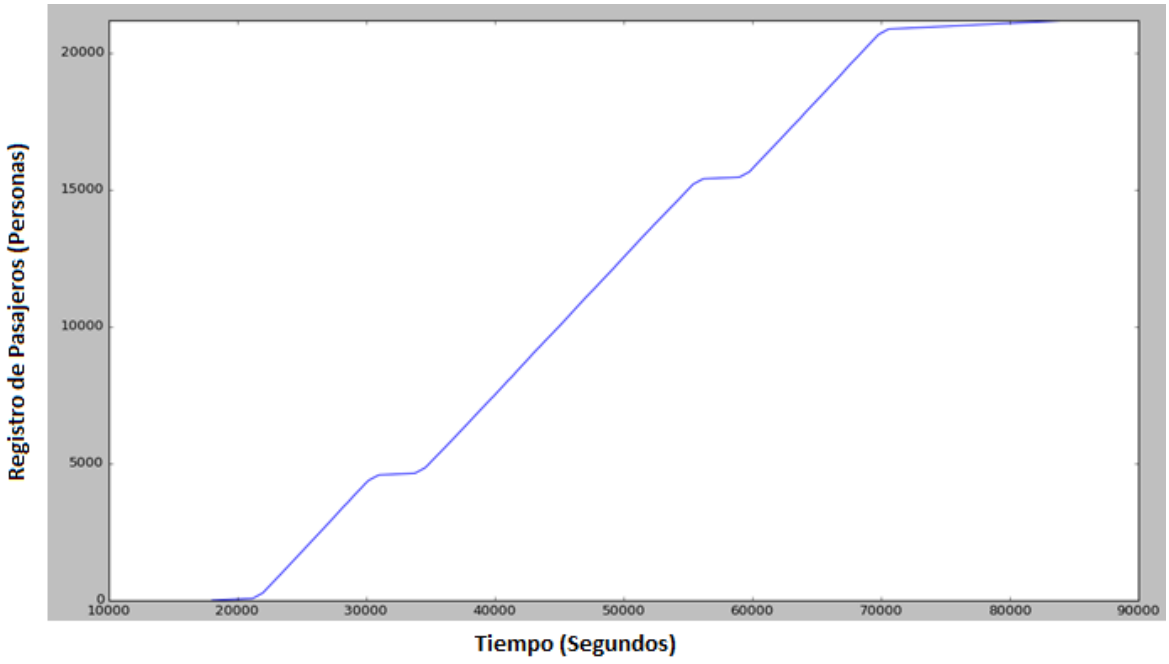

Figura 80. Comportamiento de la demanda a lo largo del día.

Como se observa en los intervalos anteriores, en hora pico la pendiente es mayor que en los intervalos en hora valle, lo que quiere decir que en la hora pico se registra la llegada de más pasajeros por segundo que en la hora valle.

La siguiente figura está compuesta por tres gráficas, las cuales representan la primera, segunda y tercera derivada de la señal de demanda de pasajeros, respectivamente:

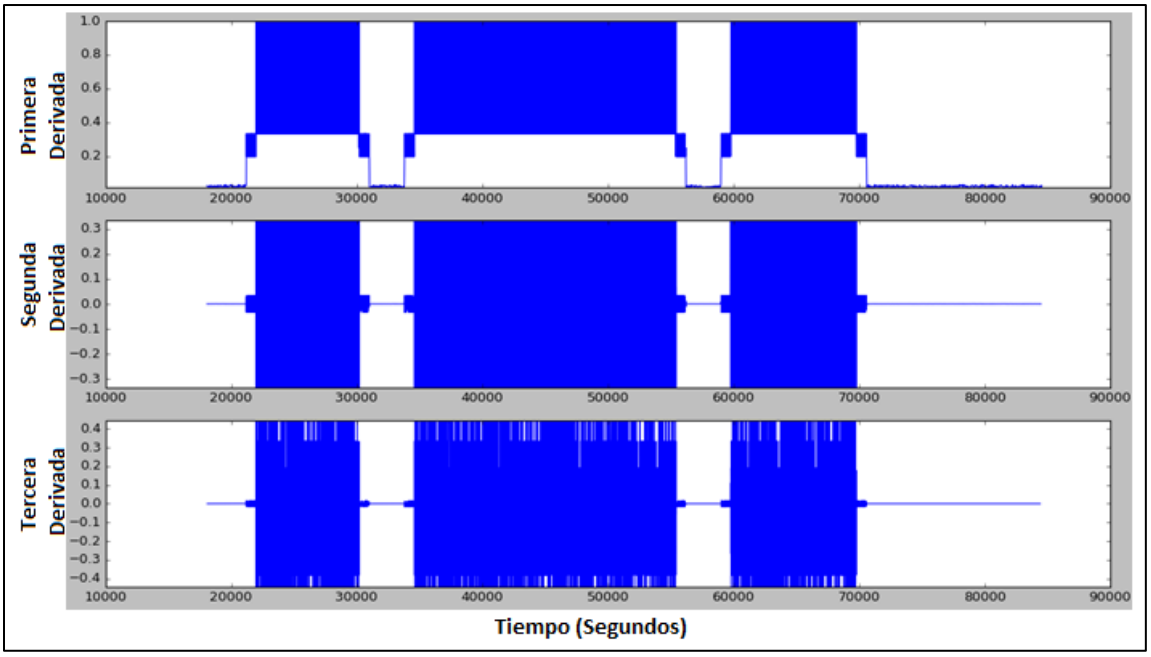

Figura 81. Primera, segunda y tercera derivada de la demanda de pasajeros

Al realizar un acercamiento a las derivadas como se observa en la siguiente imagen, se puede observar que la variación entre los valores de la franja pico de la segunda derivada son más uniformes en comparación con los de la tercera derivada, lo que permitirá que, después de aplicar la función valor absoluto y normalizar, la señal resultante sea más suave en comparación al resultado obtenido si se trabaja con la tercera derivada. Por esta razón, se trabajó con la señal obtenida de la segunda derivada:

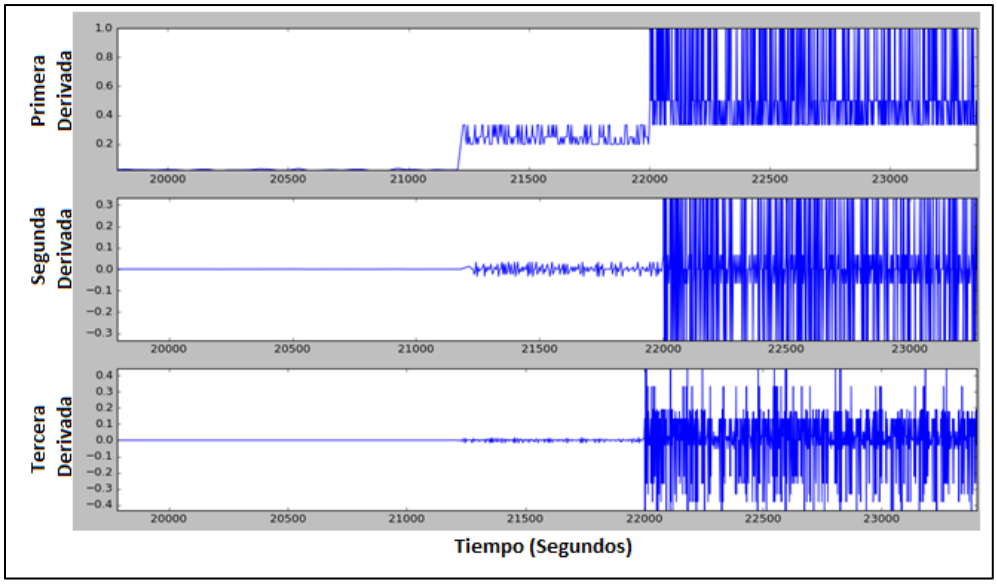

Figura 82. Zoom de las derivadas de la señal de la demanda.

Antes de clasificar las franjas horarias se filtró la señal para suavizarla, haciendo uso de un filtro de tipo Moving Average. El mejor comportamiento para el filtro se obtuvo con una ventana de 200 elementos. Primero se aplicó la función valor absoluto a la señal, después se aplicó el filtro Moving Average y posteriormente se normalizó la señal. De esta forma se obtuvo el siguiente resultado:

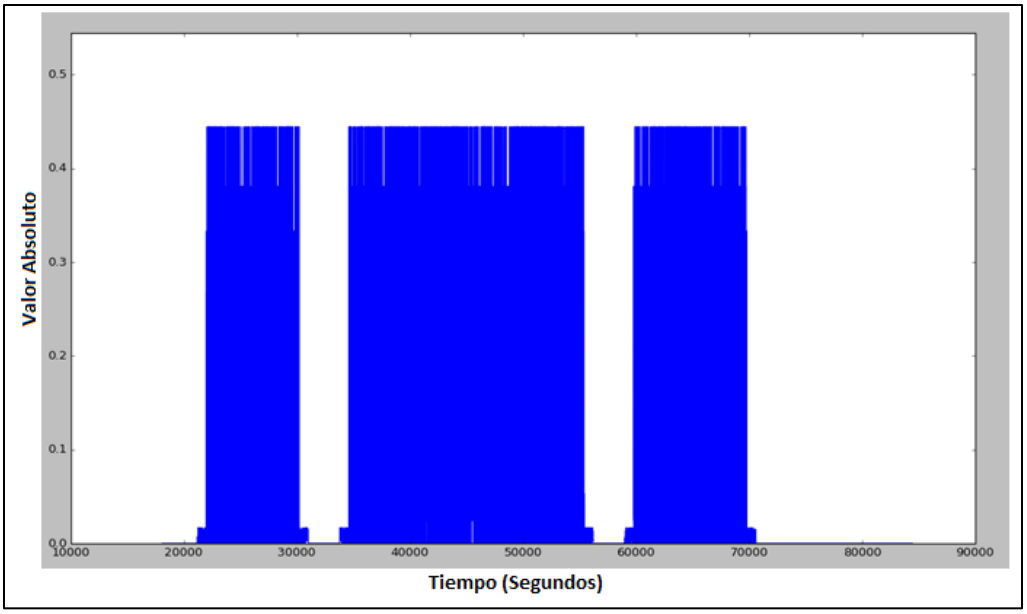

Figura 83. Valor absoluto de la segunda derivada de la señal de demanda.

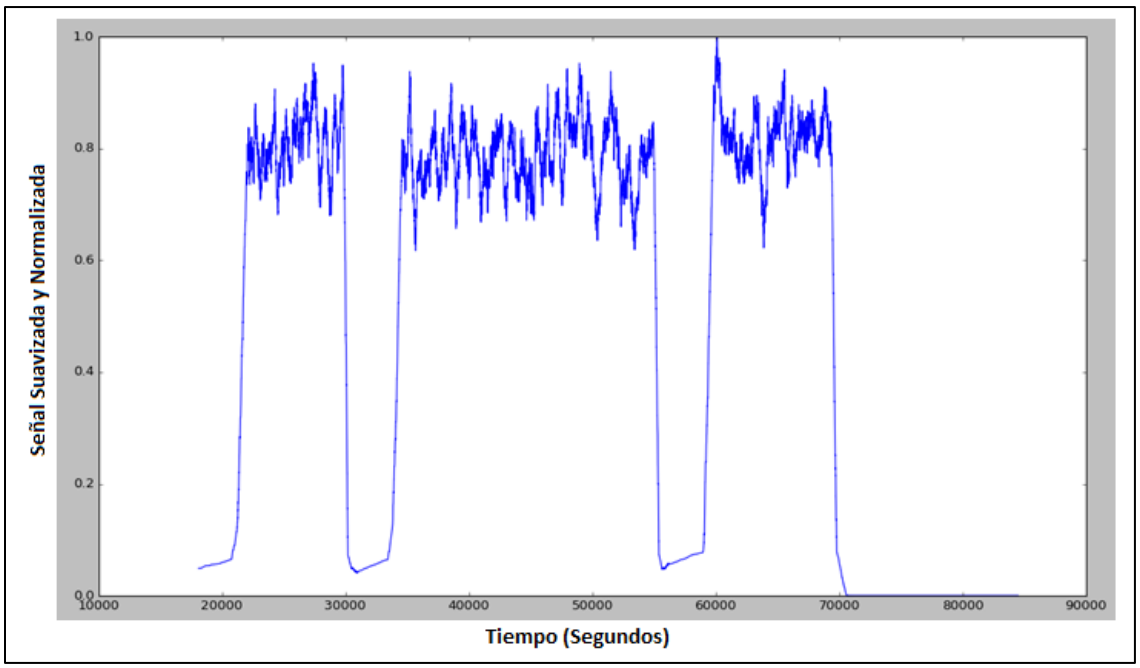

Figura 84. Segunda derivada de la señal de demanda con valor absoluto, suavización y normalización.

Como se observa, con la señal suavizada y normalizada se logran diferenciar las franjas horarias. Se escogió un umbral del 20% para discretizar la señal en dos niveles y encontrar la hora de inicio y final de franja:

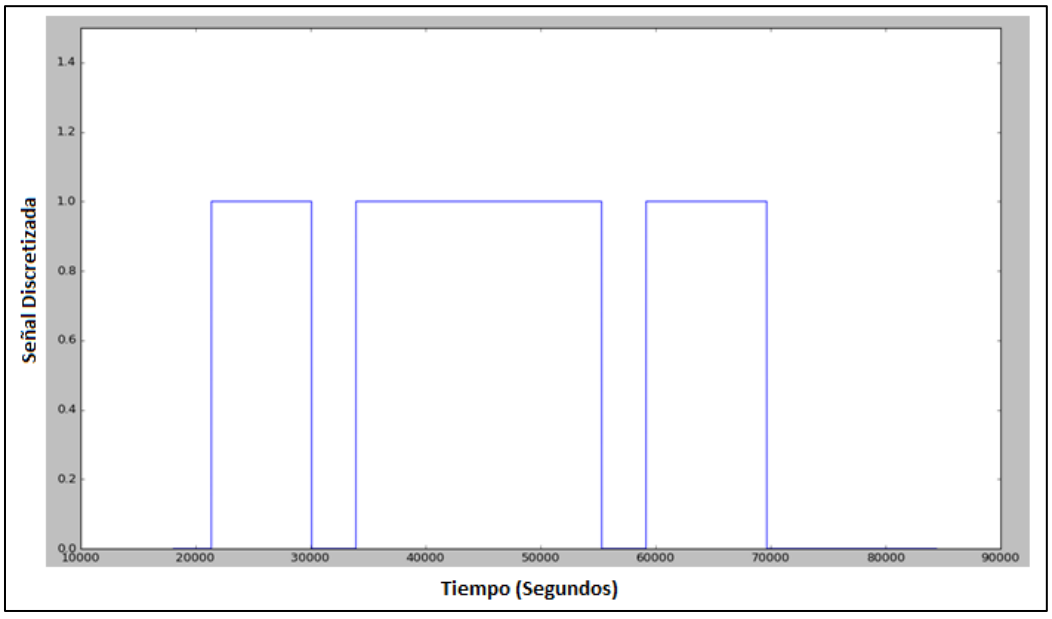

Figura 85. Señal discreta de la segunda derivada de la señal de demanda

De la señal anterior el algoritmo identifica los puntos en el dominio (tiempo) en los cuales inicia y acaba una franja horaria. Donde el valor de la señal es cero se identifica como franja horaria valle y donde el valor de la señal es uno se identifica como franja horaria pico. El algoritmo retorna un array con los intervalos hallados, como se muestra en la siguiente figura:

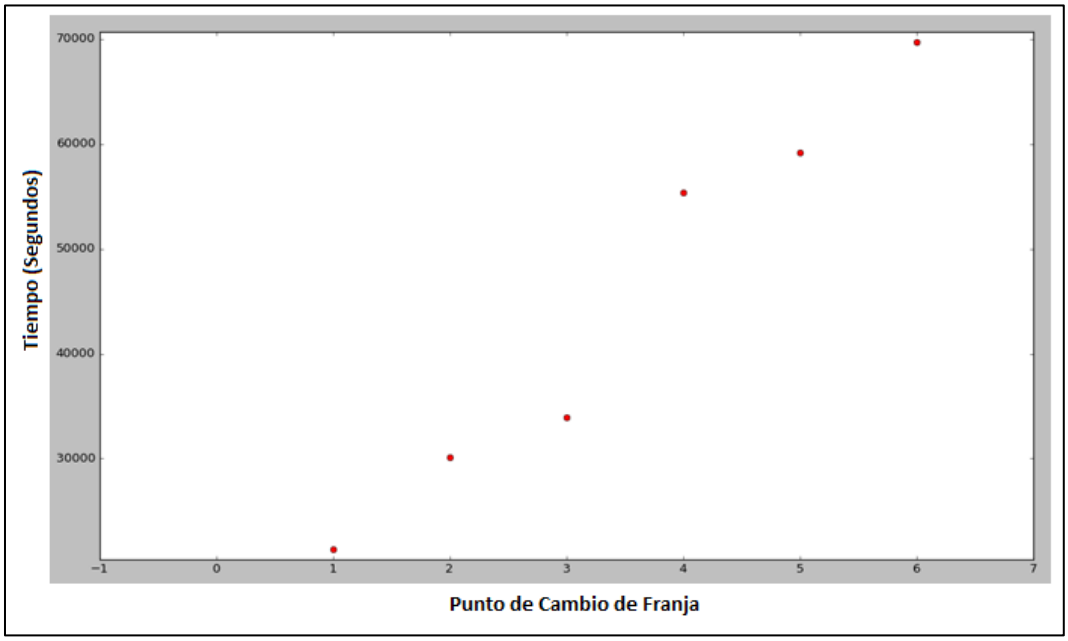

Figura 86. Intercambio de franja pico a franja valle y viceversa.

Los intervalos de tiempo, además de los entregados por el algoritmo, incluyen la hora de inicio de operación (18000 segundos) y la hora del último ingreso de un pasajero en el día (alrededor de 84600, para este caso). De esta forma el array de intervalos de tiempo fue:

[18036.0, 21389.875, 30106.5, 33963.0, 55324.0, 59185.5, 69682.5, 84639.0]

El algoritmo calcula el número de pasajeros que ingresan en estas franjas horarias para estimar el número de buses necesarios para satisfacer esta demanda. En la siguiente tabla se relacionan las franjas horarias con el número de pasajeros en cada franja (tercera columna):

Tabla 30. Asignación de pasajeros por franja horaria

Franja Horaria	Franja Horaria en segundos	Pasajeros por franja	Tiempo de Recorrido
5:00am – 5:56am	18036.0 - 21389.875	114	1232
5:56am – 8:22am	21389.875 - 30106.5	4223	1821
8:22am – 9:25am	30106.5 - 33963.0	347	980
9:25am – 3:21pm	33963.0 - 55324.0	10522	1956
3:21pm – 4:26pm	55324.0 - 59185.5	352	1613
4:26pm – 7:21pm	59185.5 - 69682.5	5122	2064
7:21pm – 11:30pm	69682.5 - 84639.0	571	1264

Para saber cuántos buses se necesitan para satisfacer la demanda se debe tener en cuenta el tiempo que tarda un bus en realizar el recorrido de cada ruta. Para realizar esto, se halla el promedio de recorrido de un bus en cada franja horaria. Este promedio se debe extraer de la tabla Despacho filtrando la fecha, la ruta y las horas que se desean analizar.

En la tabla anterior se puede observar el tiempo promedio de recorrido por cada franja horaria (columna 4). El algoritmo divide el número de pasajeros entre el tiempo de cada franja para hallar un estimado del número de pasajeros por segundo que llegan en cada franja horaria. Para hallar un valor aproximado de segundos requeridos para que lleguen 80 pasajeros a la ruta por franja horaria, se divide 80 entre el número de pasajeros por segundo hallado anteriormente. Finalmente se divide el tiempo de recorrido entre el tiempo en el que llegan 80 pasajeros a la ruta en cada franja horaria y así se obtiene el número de buses necesarios para satisfacer la demanda. El resultado será un numero decimal, así que se debe acercar a su entero mayor. El resultado del número de buses requeridos, hallado mediante el algoritmo se muestra en la siguiente tabla (columna 4):

Tabla 31. Buses requeridos por franja horaria

Franja Horaria	Franja Horaria en segundos	Pasajeros por franja	Numero de Buses
5:00am – 5:56am	18036.0 - 21389.875	114	1
5:56am – 8:22am	21389.875 - 30106.5	4223	11
8:22am – 9:25am	30106.5 - 33963.0	347	2
9:25am – 3:21pm	33963.0 - 55324.0	10522	12
3:21pm – 4:26pm	55324.0 - 59185.5	352	2
4:26pm – 7:21pm	59185.5 - 69682.5	5122	12
7:21pm – 11:30pm	69682.5 - 84639.0	571	1

Este algoritmo de optimización se diseñó para ser utilizado en una planificación a mediano plazo. El análisis anterior entrega el número de buses en distintas franjas horarias para una ruta, por lo cual deberá sumarse el número de buses que necesitan todas las rutas en una franja horaria para que la planificación sea correcta; esto partiendo de la primicia que todas las rutas compartirán los recursos (buses).

Pruebas y resultados

Para cada parte de la aplicación se llevaron a cabo un conjunto de validaciones, que nos permitieron encontrar fallas en su construcción y realizar las correcciones pertinentes.

Pruebas de la visualización histórica de datos

Para la visualización histórica de datos se usó PyGal, como había mencionado anteriormente. PyGal es una herramienta atractiva visualmente, ya que permite interactuar con las gráficas generadas. Para visualizar los datos se usó el estilo de línea, y para este estilo se puede seleccionar las curvas que se desean ver en la gráfica después de haber sido generada. Además, si se ubica el puntero del mouse sobre cada punto, PyGal genera información del punto, como el

rango, el dominio y la curva a la que pertenece el punto, haciendo más cómoda la visualización de los datos.

Partiendo del mismo algoritmo usado para crear una señal de demanda que se comporte de acuerdo a las franjas horarias pico y valle propuestas por TransMilenio, se diseñó un algoritmo para llenar las bases de datos. Este algoritmo llena las tablas Despacho, Usuario y TiempoAsignacion, las cuales contienen la información acerca de los indicadores KPI y se llenan a medida que ingresan pasajeros a la zona de espera o cuando se despachan buses alimentadores. Por supuesto, el ingreso de pasajeros también se simula por medio del algoritmo.

Debido al procesamiento que se requiere para realizar la búsqueda de datos e importarlos para su visualización, se decidió crear datos para 12 días: 3 días hábiles, 3 domingos, 3 sábados y 3 días festivos. Los datos se crearon desde las 4:30 pm hasta las 7:30 pm.

En la actualidad el sistema de asignación de buses a las rutas alimentadoras se hace con base en una frecuencia determinada. El objetivo de la propuesta en este proyecto es que la asignación sea de acuerdo a la demanda de pasajeros. Por esta razón el algoritmo llena las tablas de la base de datos en función de la demanda de pasajeros. Como lo explican los diagramas de flujo de llenado de la base de datos.

Las siguientes gráficas muestran un arquetipo de la gestión de buses alimentadores. En el modo de búsqueda “Rango” se escogió un rango para visualizar la información del día 20 de mayo.

En la siguiente gráfica se observa el comportamiento de los pasajeros en espera de las tres rutas. Se puede observar la diferencia en el flujo de pasajeros de cada ruta, donde la llegada de pasajeros a la ruta 83 es mayor para esta franja horaria. También se observa que en esta franja horaria en la ruta 83 se alcanza el máximo de pasajeros en espera y disminuye mucho antes de que disminuya el número de pasajeros en espera en las otras rutas, ya que los dos criterios de

asignación de bus son que se cumpla un tiempo máximo de espera o que la densidad de pasajeros llegue al límite.

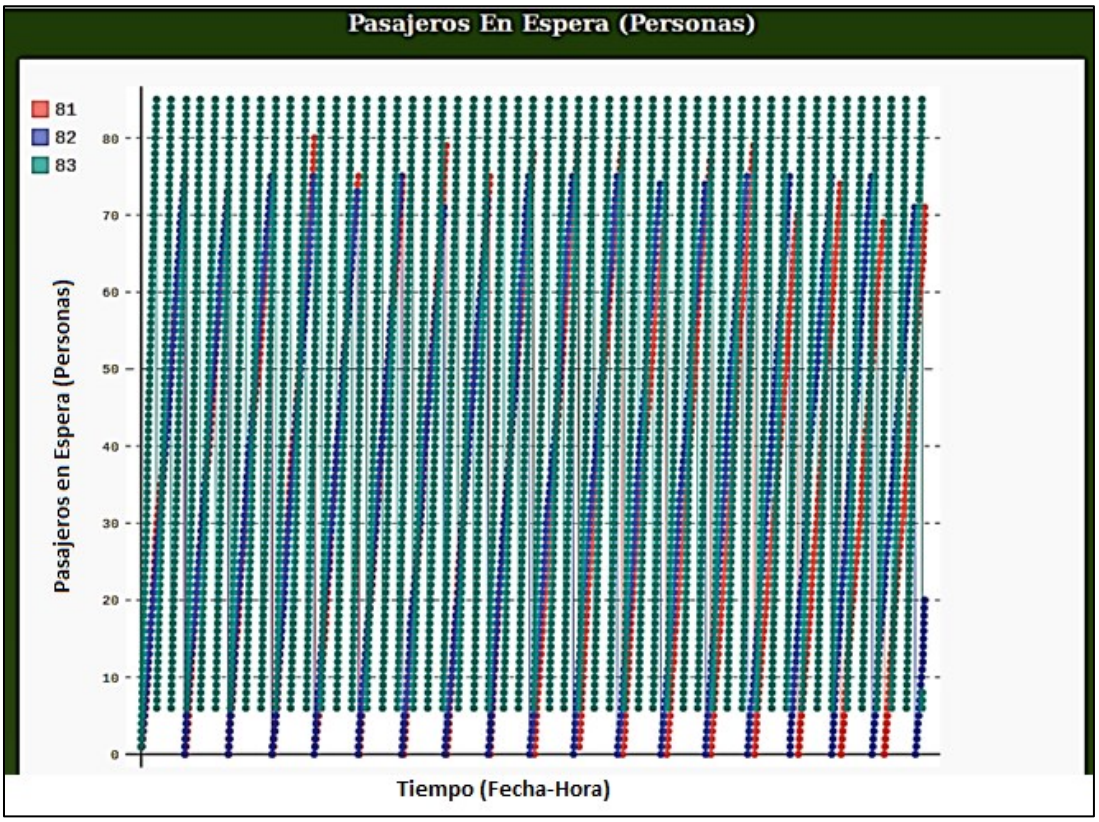

Figura 87. Visualización de datos de pasajeros en espera en las tres rutas de la estación

En la siguiente figura se observa la carga de pasajeros en donde se corrobora la información anterior, donde la ruta 83 usa los recursos eficientemente y los buses son utilizados en su capacidad máxima. La carga de pasajeros de las rutas 82 y 81 no es tan eficientes en la ruta 83, pero esto responde a los tiempos máximos de espera entre despachos.

Estas gráficas permitirían evaluar la capacidad de los buses y tomar decisiones acerca del tamaño de los buses. Se puede concluir que, para esta franja horaria, la ruta 83 usa más los recursos, y que la ruta 82 no usa la capacidad máxima de los buses:

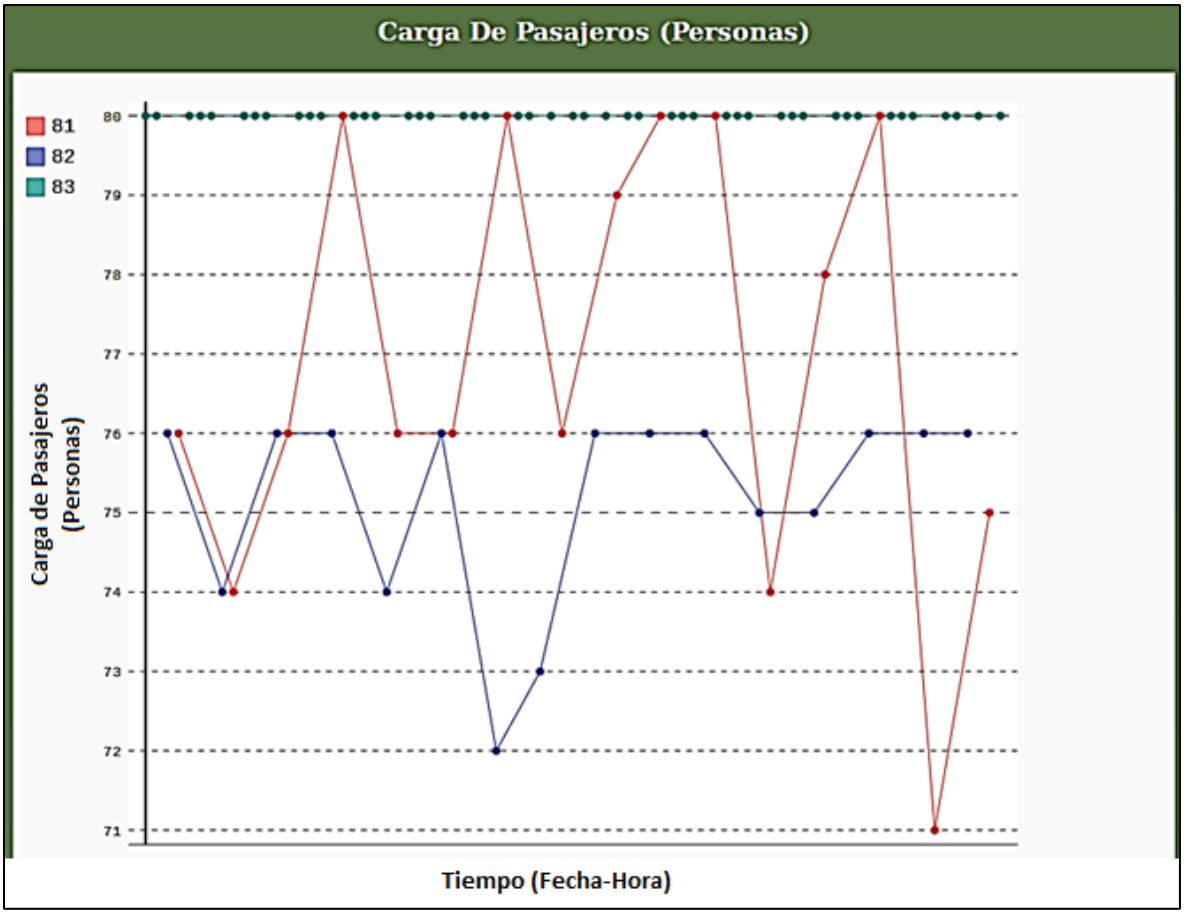

Figura 88. Visualización de datos de carga de pasajeros en las tres rutas de la estación

En la siguiente figura se encuentra la visualización de los datos de Delay, que corresponde a los tiempos que se retrasan los buses en hacer los recorridos. Estos datos permitirán hacer una planificación a mediano plazo para reasignar los buses en las distintas franjas horarias. En la gráfica se observa que en esta franja horaria pueden existir retrasos de hasta 10 minutos:

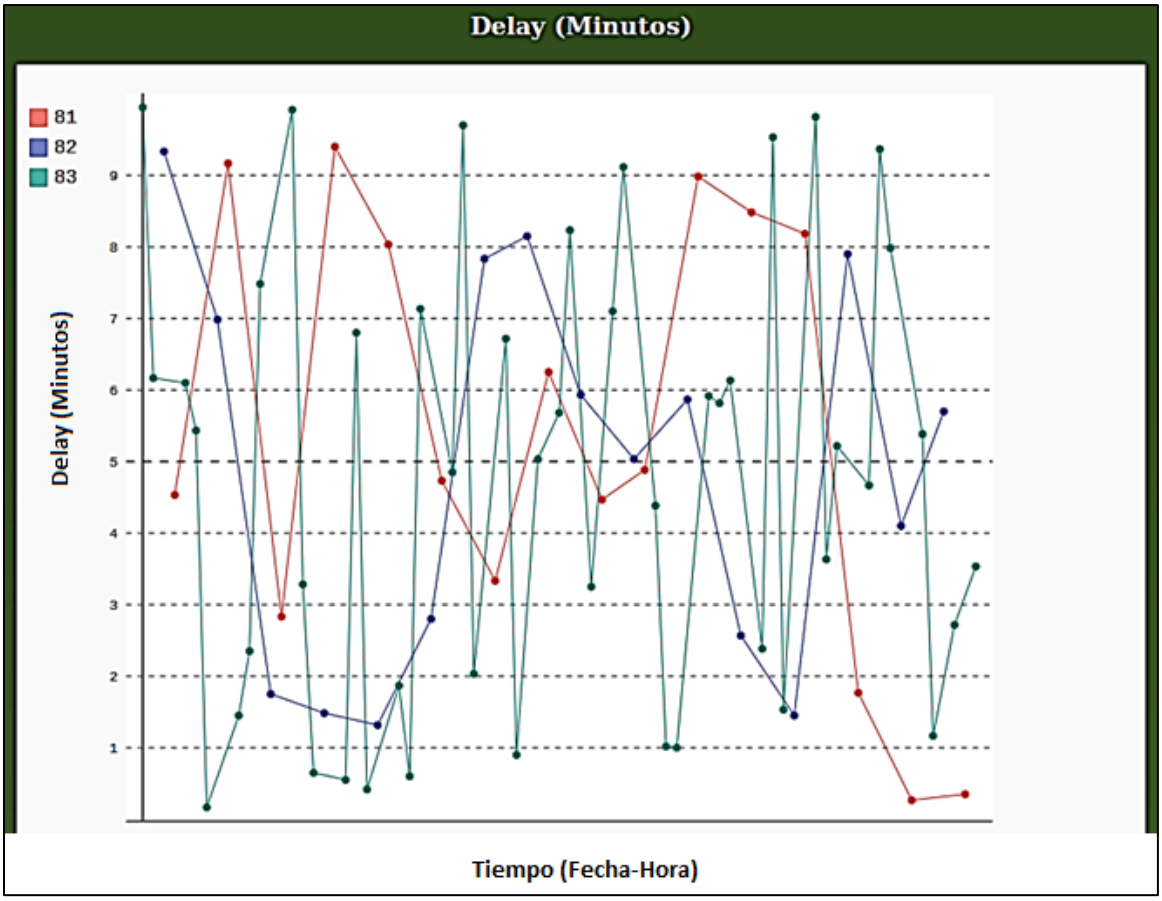

Figura 89. Visualización de datos de Delay en las tres rutas de la estación

El indicador Factor de Confiabilidad permitirá evaluar los tiempos de recorrido de los buses. Este se verá afectado por la carretera de la ruta, el tráfico y los tiempos de bajada de los pasajeros. Un análisis de este factor podría decantar en mejorar las rutas de los alimentadores. En este caso el indicador no es el que se espera del gestor, y esto se debe a que el algoritmo se implementó para que el Delay fuera aleatorio como se observa en la figura anterior. El factor de confiabilidad puede verse afectado por un accidente en la vía, lo que permitiría evaluar que tan preparado está el sistema para afrontar estos riesgos:

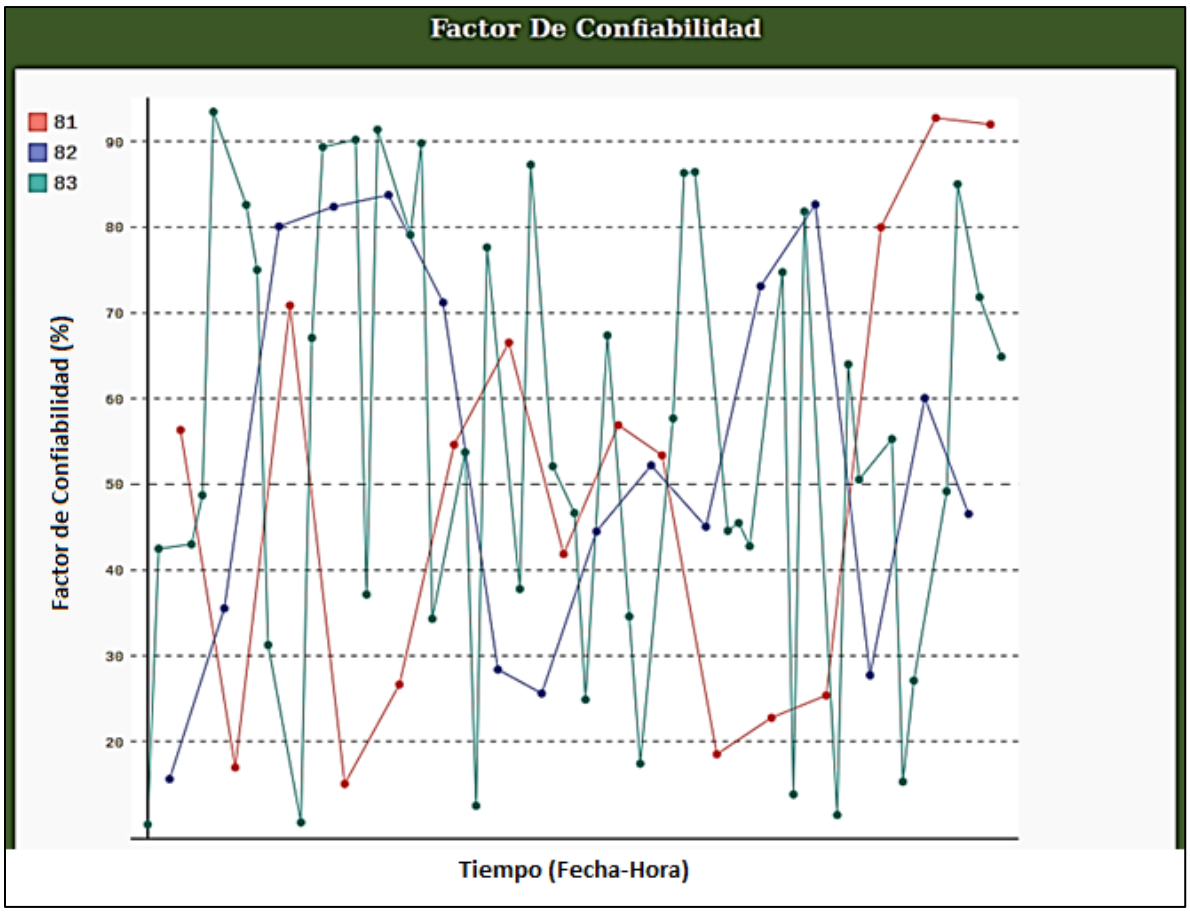

Figura 90. Visualización de datos del Factor De Confiabilidad en las tres rutas de la estación

En la siguiente gráfica se observa el comportamiento del Tiempo de Espera. Este se verá afectado por la demora del operador en asignar el bus a la ruta luego de una solicitud de asignación de bus. También se verá afectado por la existencia de buses disponibles, si en el momento de una solicitud de asignación no hay buses disponibles, este tiempo será mucho mayor. En la siguiente figura se ve el comportamiento esperado, en donde la asignación no es mayor de 1 minuto:

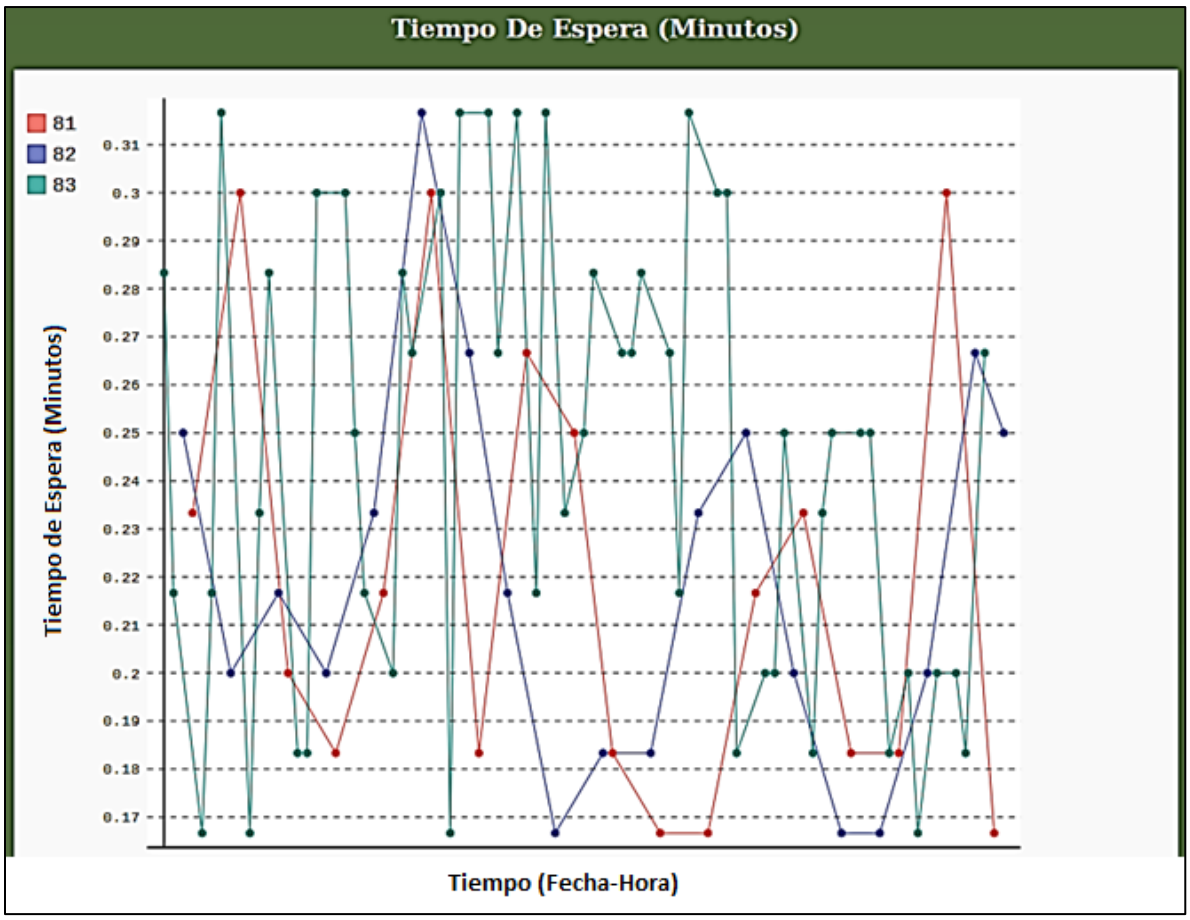

Figura 91. Visualización de datos de los Tiempos de Espera en las tres rutas de la estación

Seleccionando la opción de comparación, se compararon 4 días de diferentes tipos de un mes (día hábil, festivo, sábado y domingo) en la ruta 81. Se escogieron los siguientes días:

- 2 de mayo: Festivo.
- 14 de mayo: sábado.
- 15 de mayo: Domingo.
- 20 de mayo: Día hábil.

En la opción de comparación se visualiza una gráfica por cada ruta y cada indicador. De las gráficas se analiza la variación de la demanda de pasajeros y sus consecuencias en distintos días. A continuación, se muestran los resultados obtenidos para esta comparación:

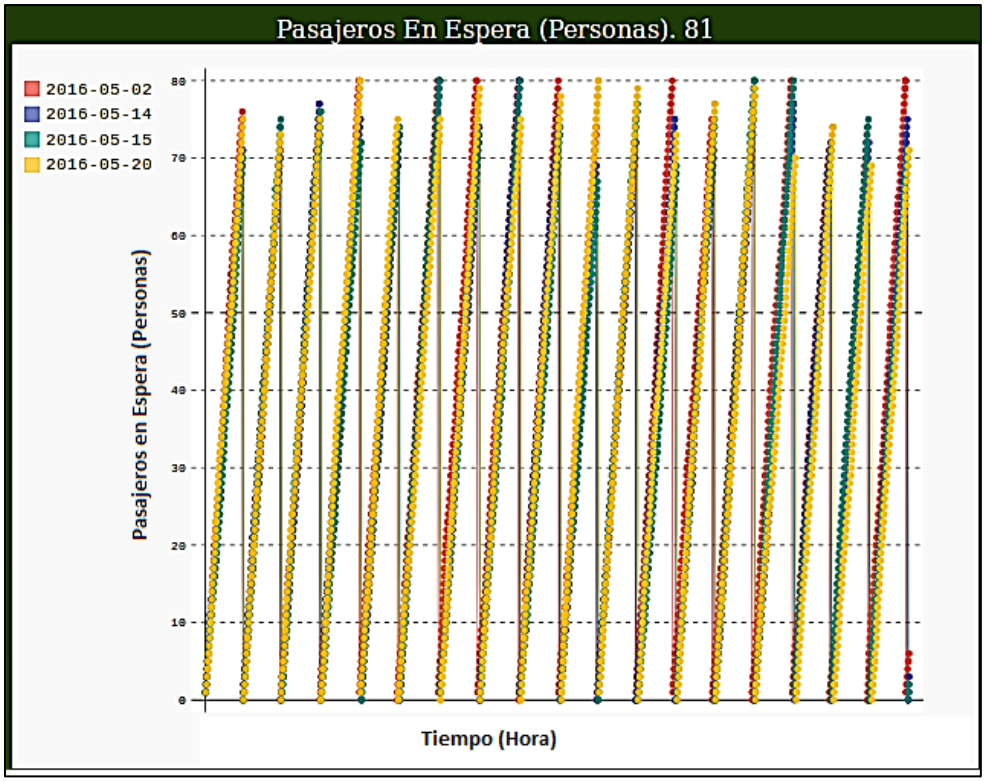

Figura 92. Visualización de datos de Pasajeros en Espera de 4 días en la ruta 81

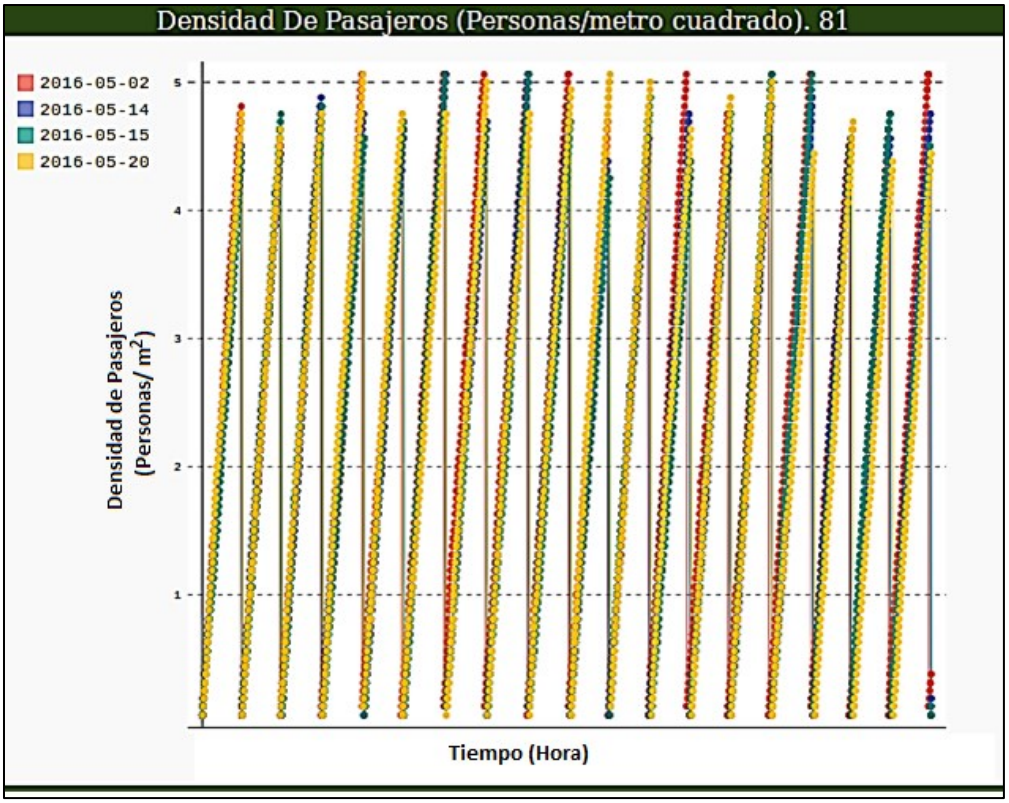

Figura 93. Visualización de datos de Densidad de Pasajeros de 4 días en la ruta 81

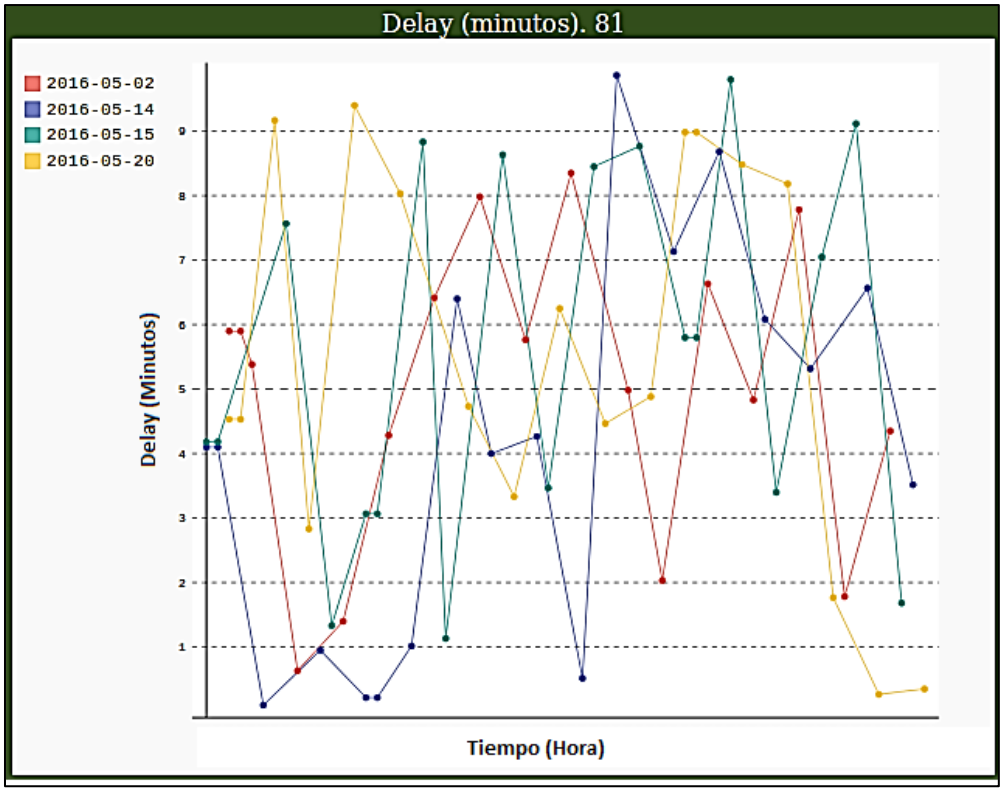

Figura 94. Visualización de datos de Delay de 4 días en la ruta 81

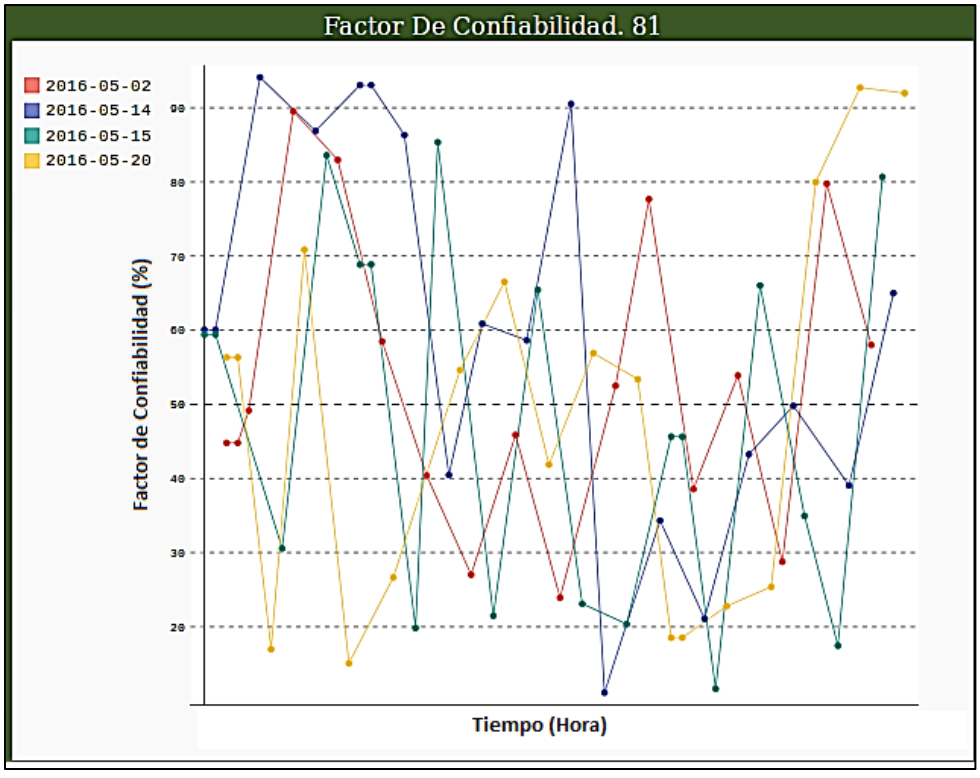

Figura 95. Visualización de datos de Factor de Confiabilidad de 4 días en la ruta 81

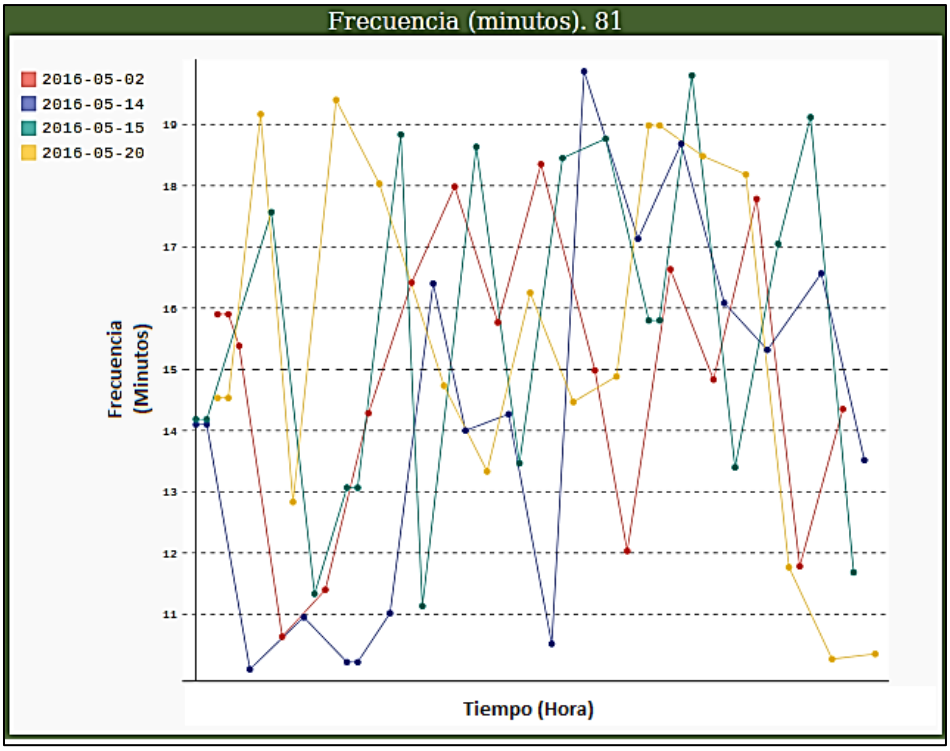

Figura 96. Visualización de datos de Frecuencia de 4 días en la ruta 81

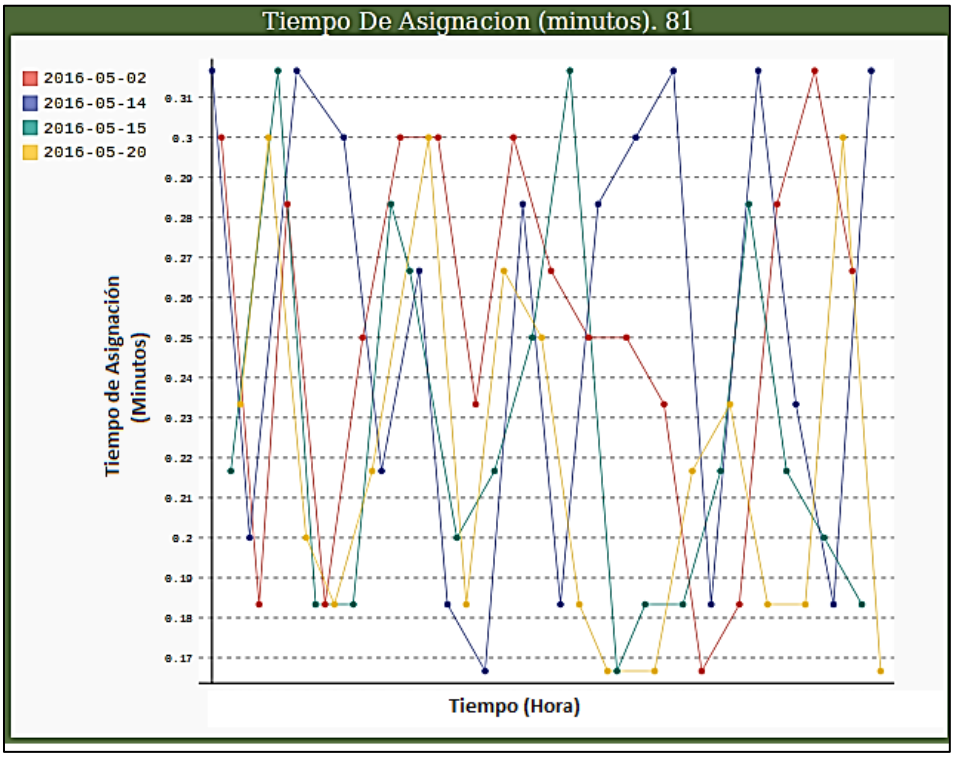

Figura 97. Visualización de datos de Tiempo de Asignación de 4 días en la ruta 81

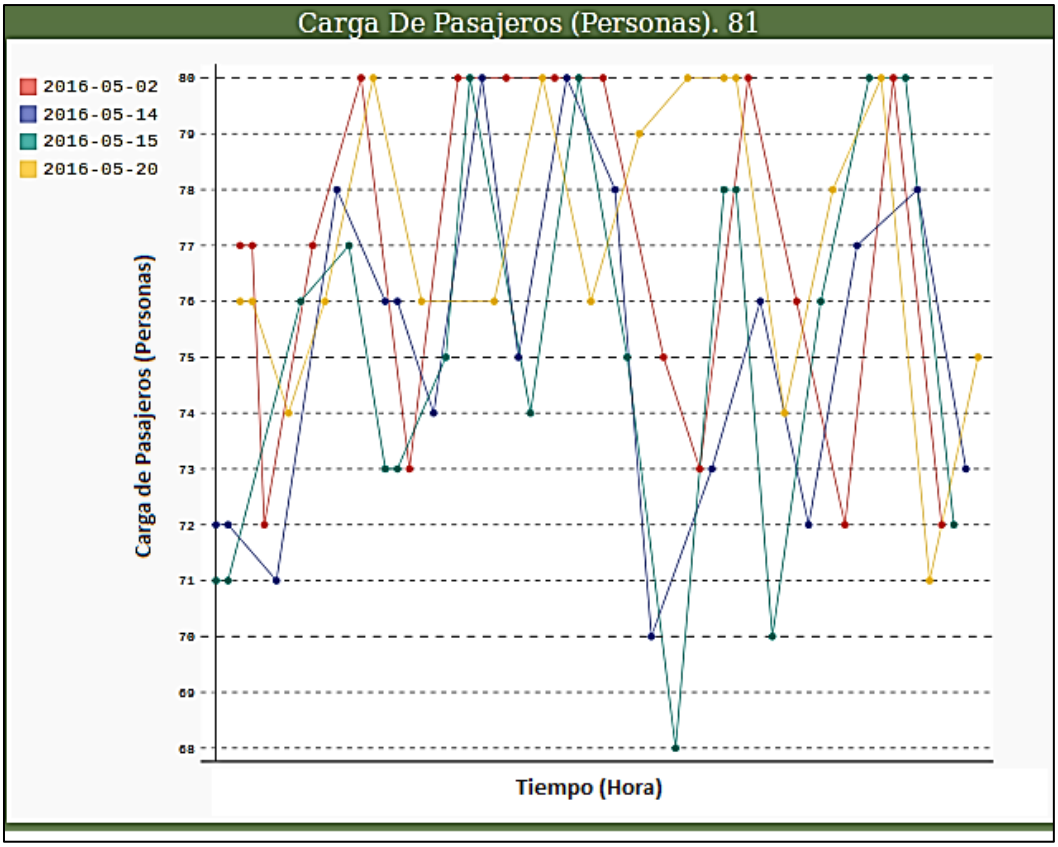

Figura 98. Visualización de datos de Carga de Pasajeros de 4 días en la ruta 81

Comparando distintos días se podría analizar la variación de la demanda en distintas temporadas del año, o entre temporadas de vacaciones y temporadas laborales. Hablaremos de más de los posibles análisis de estas comparaciones más adelante.

Pruebas del algoritmo

El algoritmo fue probado simulando diferentes comportamientos de demandas. En las siguientes graficas se muestran las pendientes altas como franjas horarias pico y las pendientes bajas como franjas horarias valle. En la siguiente imagen se observa la señal de comportamiento de demanda simulada:

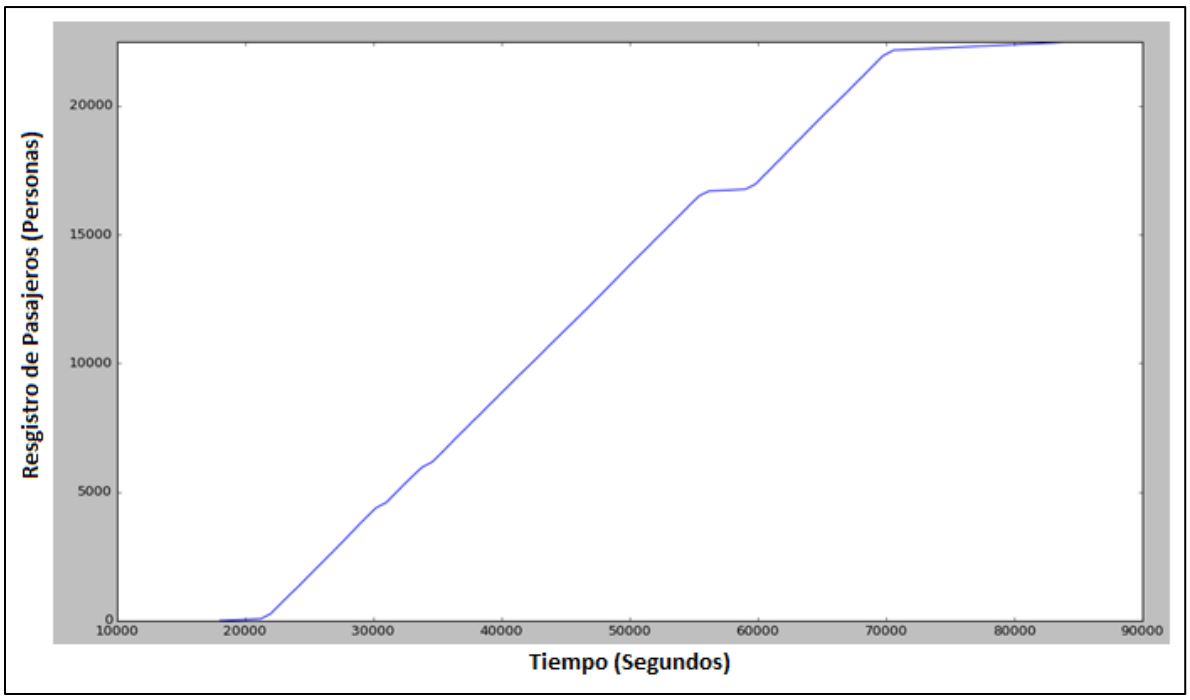

Figura 99. Primera señal de demanda simulada para prueba de algoritmo.

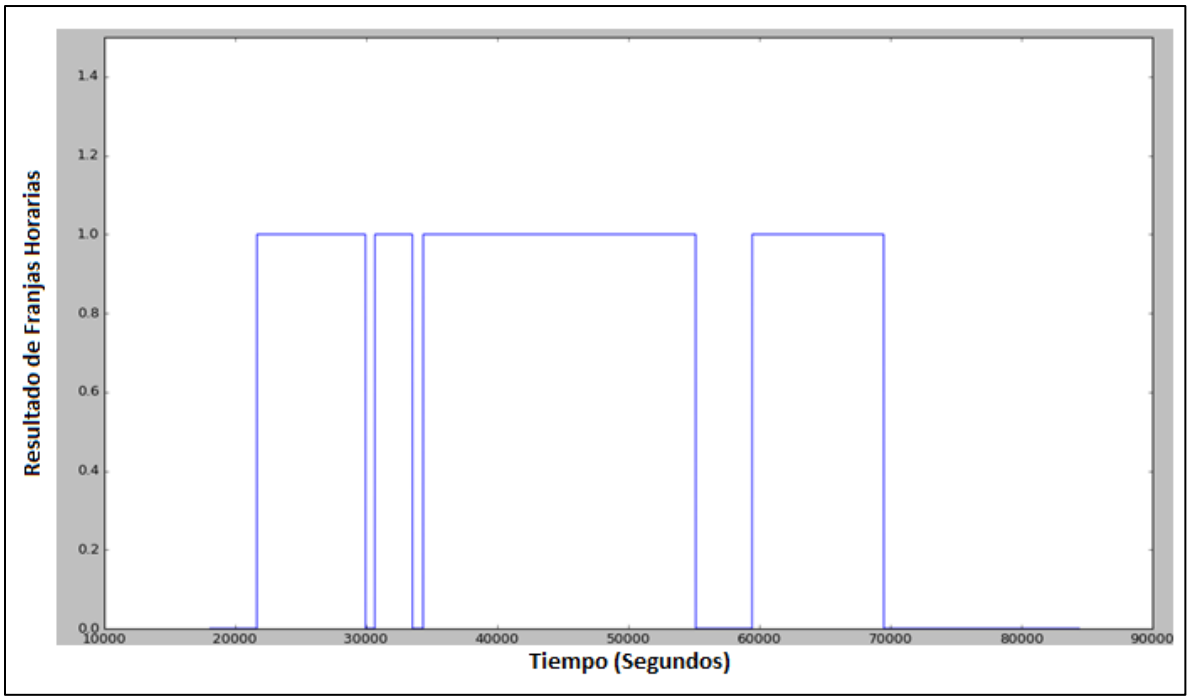

Figura 100. Discretizacion de las franjas horarias pico y valle de la primera prueba.

El resultado para la primera simulación y el cálculo de las franjas horarias y buses respectivos es el siguiente:

Tabla 32. Cálculo de franjas horarias y buses requeridos, primera prueba del algoritmo

Franja Horaria	Franja Horaria en segundos	Pasajeros por franja	Numero de Buses
5:00am – 6:01am	18033.0 - 21669.25	183	1
6:01am – 8:19am	21669.25 - 29953.375	4075	12
8:19am – 8:31am	29953.375 - 30671.625	204	5
8:31am – 9:18am	30671.625 - 33537.375	1334	12
9:18am – 9:32am	33537.375 - 34353.75	267	7
9:32am – 3:19pm	34353.75 - 55153.375	10276	13
3:19pm – 4:30pm	55153.375 - 59458.125	494	2
4:30pm – 7:18pm	59458.125 - 69504.5	4948	8
7:18pm – 11-30pm	69504.5 - 84612.0	673	1

Sera una decisión administrativa si en las pequeñas franjas valle, resaltadas con amarillo, se asignan los buses propuestos o se toman estas franjas junto con las franjas pico resaltadas en azul para formar una sola franja pico con 12 buses asignados. A continuación, se probó el algoritmo con otra señal creada a partir del simulador. Se diseñó la siguiente señal para evaluar el algoritmo:

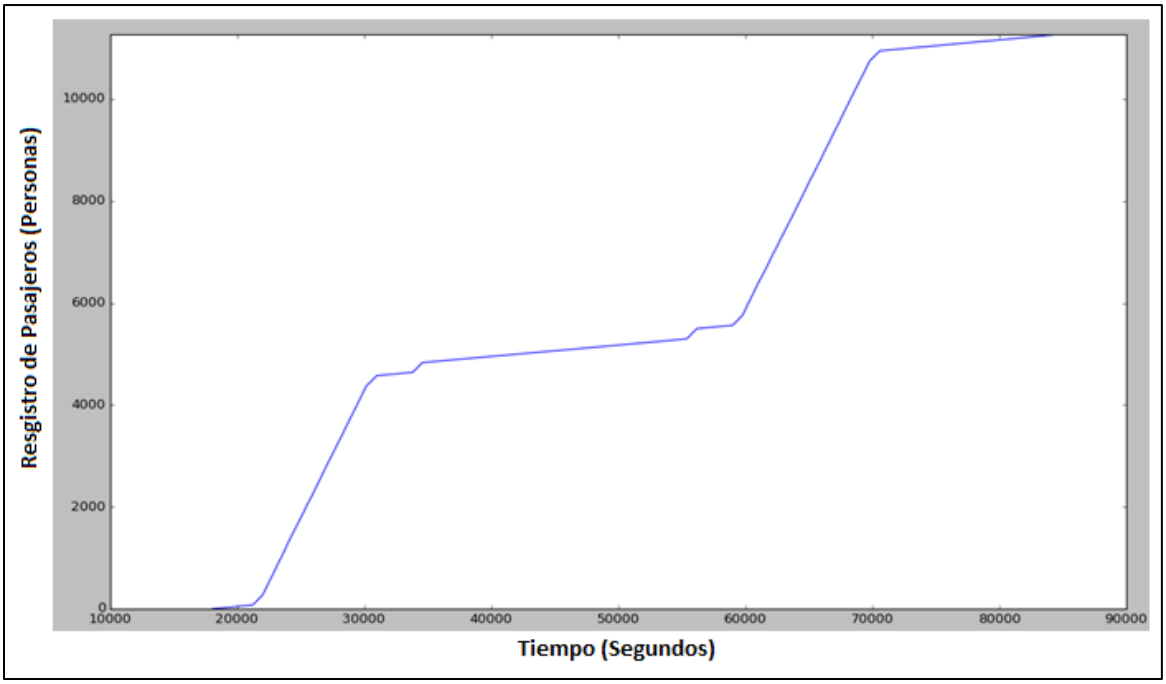

Figura 101. Segunda señal de demanda simulada para prueba de algoritmo

El resultado para la segunda simulación y el cálculo de las franjas horarias y buses respectivos es el siguiente:

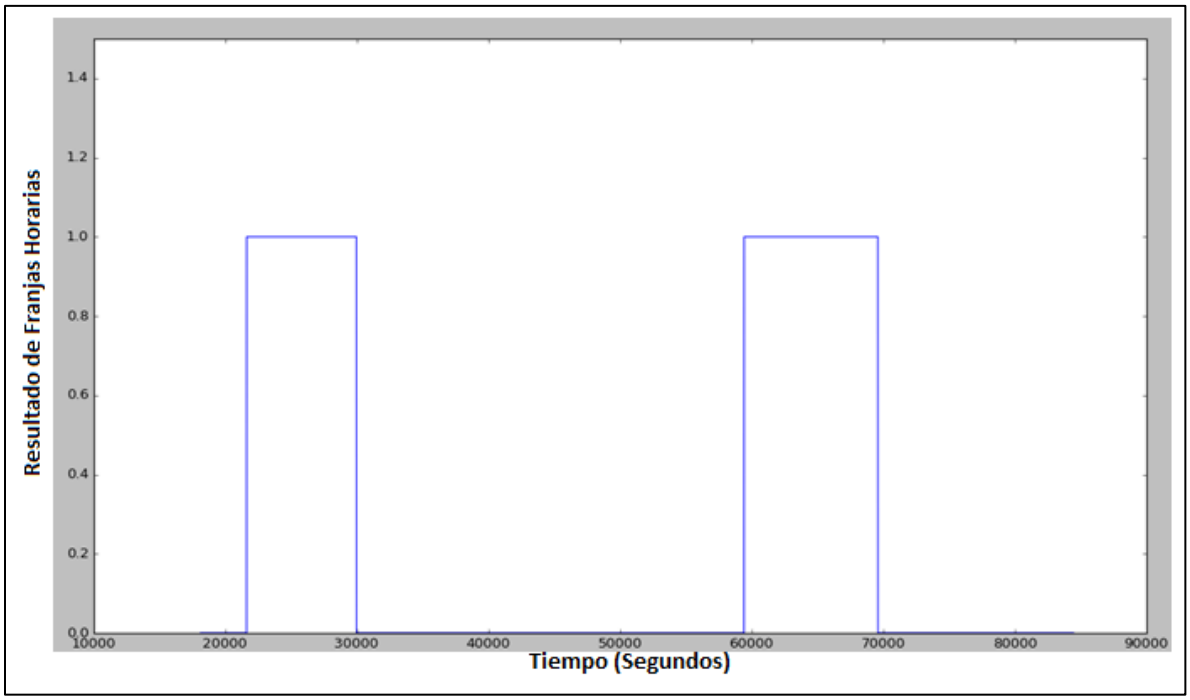

Figura 102. Resultado de la discretizacion de las franjas horarias pico y valle de la segunda prueba

Tabla 33. Cálculo de franjas horarias y buses requeridos, segunda prueba del algoritmo

Franja Horaria	Franja Horaria en segundos	Pasajeros por franja	Numero de Buses
5:00am – 6:00am	18054.0 - 21635.75	168	1
6:00am – 8:19am	21635.75 - 29983.75	4100	12
8:19am – 4:30pm	29983.75 - 59429.875	1391	1
4:30pm – 7:19pm	59429.875 - 69598.5	4978	12
7:19pm – 11:30pm	69598.5 - 84646	629	1

Conclusiones

La recolección de datos sobre el comportamiento de la demanda de pasajeros de un sistema de transporte masivo converge en numerosas posibilidades de análisis de los mismos; de cada subsistema de los que se recolecten datos se puede obtener información suficiente para tomar decisiones acerca de la administración de los recursos de la operación.

La aplicación entrega un conjunto de información que aporta un valor relevante para analizar el comportamiento del sistema. Tal es el caso de los indicadores de calidad de servicio propuestos en este proyecto, los cuales permiten planificar el servicio a corto y mediano plazo y entregan información de gran utilidad para evaluar los costos tanto del operador como el usuario, y con esto, mejorar la calidad del servicio prestado. En este sentido, los indicadores pueden hacer parte integral de plan de mejoramiento y evaluación, que busque cuantificar el desempeño del sistema e identificar sus fortalezas y debilidades, con el fin de sugerir mejorías y tomar acciones correctivas. Además, su procesamiento puede permitir la creación de nuevos indicadores de desempeño, como puede ser el número de buses necesarios para satisfacer la demanda de pasajeros de des alimentación garantizando buenos niveles de calidad de servicio.

Basados en el caso de estudio trabajado sobre el sistema Transmilenio, actualmente se mide el Índice de Pasajeros por Kilómetro (IPK), que permite medir los costos en que incurre el operador en la prestación del servicio, pero este es un indicador de desempeño, mas no de calidad del servicio, y no se tiene en cuenta ningún indicador que proporcione información para medir los costos de viaje de los pasajeros y la calidad del servicio en función de factores como el tiempo invertido por cada usuario en el uso del sistema (tiempo de espera de la ruta, tiempo de recorrido, tiempo de transferencia inter-modal).

Los buses alimentadores transitan sobre carriles mixtos, compartiendo la vía con el tráfico vehicular particular, viéndose expuestos a externalidades negativas. Por esta razón, es importante minimizar los costos para el operador y los costos de viaje de los pasajeros, aminorando los efectos de las externalidades que en cierta medida pueden ser perjudiciales para la operación, como pueden ser, la frecuencia de salida de los buses alimentadores y los tiempos de espera promedio de los usuarios de las rutas alimentadores. El método de despacho de buses actual no garantiza unos costos de viaje a los pasajeros suficientemente bajos, ya que los intervalos de las rutas alimentadoras están programados por una frecuencia determinada basada en parámetros operacionales aplicados a largo plazo, sin tener en cuenta algunos factores que influyen en la demanda de pasajeros actual de cada ruta alimentadora.

USO DE TECNOLOGÍAS CLAVES: El manejo de la información a través de una base de datos relacional y un servidor web proporciona uniformidad en la toma de decisiones y en la aplicación de políticas y métodos para la gestión de estos datos. La implementación de un modelo de gestión que responda a la demanda de pasajeros mediante el análisis de los datos esta en concordancia con las proyecciones del gobierno actual en temas de datos abiertos.

Los recursos tecnológicos de almacenamiento y procesamiento de esta información deben ser evaluados desde las perspectivas de Big Data y Fog Computing, con miras a satisfacer la cantidad de datos involucrados, debido a que cada módulo de una ruta alimentadora podrá entregar datos en intervalos mínimos de tiempo de hasta 2 segundos en horas pico.

El uso de software de simulación como VISSIM, utilizado en este proyecto, puede ayudar a analizar mejor las problemáticas actuales incorporadas a los sistemas de transporte masivo BRT, que requieren implementar soluciones eficientes y económicas, sin perjudicar la prestación del servicio.

PERSPECTIVA ECONÓMICA: Analizando los resultados desde la perspectiva económica actual, observando las gráficas del caso de estudio seleccionado, se puede observar que para una ruta de la estación banderas un gran número de los despachos de buses se realizaron con una baja cantidad de usuarios en relación a su capacidad, lo cual genera costos adicionales al operador. Por otro lado, se observa que las estaciones Kennedy Hospital y Timiza tienen una mayor demanda de pasajeros, generando un incremento constante de la cantidad de usuarios en espera, en parte debido a que los buses fueron configurados en la simulación para no sobrepasar su capacidad nominal. En este caso, es el costo de viaje de los usuarios es el que se ve afectado, bien sea por la calidad del servicio, como por el tiempo que deben esperar para acceder a un bus.

De acuerdo a la información entregada por los ingenieros del área operativa y técnica del Sistema Transmilenio, sobre los procedimientos técnicos de la programación de rutas alimentadoras, el contrato por encuestas de movilidad del sistema cuesta alrededor de 6000 millones de pesos y se realiza cada 5 años. Los resultados de esta encuesta pierden fiabilidad con el tiempo y son propensos al error humano en la adquisición y digitalización de los mismos. La implementación de este módulo para la recolección de datos permitiría aumentar la confiabilidad en los mismos sin mayores gastos en infraestructura. En el caso de la integración y adaptación de las tarjetas RFID que operan actualmente en la mayoría de los sistemas BRT a la lectura de los módulos de gestión ubicados en las plataformas de rutas alimentadoras es sencilla y no requieren cambios significativos dentro de su configuración.

Actualmente los operadores no están obligados a aumentar el número de flotas para operar en cada ruta alimentadora, incluso si en primera vista se observa que es necesario, con el modelo de adquisición de datos se podría cambiar este aspecto, ya que los requerimientos estarían basados en datos certeros y actualizados.

Un modelo de gestión de buses alimentadores como el propuesto permitiría que se generen nuevos proyectos de investigación alrededor del análisis de datos entregados por el modelo. Se podría comparar el comportamiento del sistema con otros datos obtenidos como el comportamiento climático o la densidad poblacional de una zona específica.

El uso de ITS es un factor clave en el mejoramiento de los servicios de transporte dentro de los ambientes urbanos, pero debe ir de la mano de una adecuada infraestructura y administración de los recursos. Los usuarios también juegan un rol importante en el aprovechamiento y buen uso de estas tecnologías, siendo partícipes activos del proceso bajo el cumplimiento de normas básicas de comportamiento.

TRABAJOS FUTUROS: Como trabajos posteriores, el modelo puede ser mejorado ofreciendo una adquisición de los datos más transparente para el usuario y brindándole una mejor experiencia dentro del sistema. La tecnología RFID podría funcionar para lograr esto, pero siempre se debe buscar educar a los pasajeros para que entiendan que estos procesos logísticos mejoran la gestión de los recursos del sistema y repercuten positivamente en los costos de los pasajeros.

Referencias

Assaf, M. H., & Williams, K. M. (2011). RFID for optimisation of public transportation system. *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2011 Seventh International Conference on*, 407–412. http://doi.org/10.1109/ISSNIP.2011.6146604

Bash, E. (2015). *No Title No Title*. *PhD Proposal* (Vol. 1). http://doi.org/10.1017/CBO9781107415324.004

Ca??on-Lozano, Y., Melo-Castillo, A., Gomez-Perilla, C. A., Banse, K., & Herrera-Quintero, L. F. (2013). Web service platform for automatic generation of O/D matrix for mass transportation systems. *2013 13th International Conference on ITS Telecommunications, ITST 2013*, 462–467. http://doi.org/10.1109/ITST.2013.6685589

Coskun, V., Ok, K., & Ozdenizci, B. (2012). *Near Field Communication: From Theory to Practice*.

Dunant, H. (2013). *Libro Blanco sobre la aplicación de la Tecnología NFC en el Transporte Público*.

Element14. (2016). EXPLORE-NFC Software Download. Retrieved from https://www.element14.com/community/docs/DOC-65447/l/explore-nfc-software-and-project?ICID=knode-devkitnfc-quick

Fereira, M. L., Marte, C. L., Medeiros, J. E. L. D. E., Sakurai, C. A., Fontana, C. F., Paulo, P. D. S., … Paulo, U. D. S. (2009). RFID for Real Time Passenger Monitoring OF TRANSPORTATION PASSENGERS PUBLIC, 170–175.

Ferreira, M. L., Gouveia, J. A. M. De, Facchini, E., Pokorny, M. S., & Dias, E. M. (2012). Real time monitoring of public transit passenger flows through Radio Frequency Identification - RFID technology embedded in fare smart cards, *II*.

Finzgar, L., & Trebar, M. (2011). Use of NFC and QR code identification in an electronic ticket system for public transport. *SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks*.

Gautam, J., Kumar, Y., & Gupta, A. (2014). Existing scenario of near field communication in transport sector. *2014 International Conference on Signal Processing and Integrated Networks (SPIN)*, 327–332. http://doi.org/10.1109/SPIN.2014.6776972

Guide, Q. S. (2015). AN11480 Quick Start-up Guide for EXPLORE-NFC working with Raspberry Pi, (February).

Hacks, C. (2016a). Raspberry Pi to Arduino Shields Connection Bridge. Retrieved from https://www.cooking-hacks.com/documentation/tutorials/raspberry-pi-to-arduino-shields-connection-bridge

Hacks, C. (2016b). RFID 13.56 MHz / NFC Module for Arduino and Raspberry Pi. Retrieved from https://www.cooking-hacks.com/documentation/tutorials/rfid-13-56-mhz-nfc-module-arduino-raspberry-pi-tutorial/

Hamilton, P., & Sankaranarayanan, S. (2013). Intelligent Agent Based RFID System for on

Demand Bus Scheduling and Ticketing. *International Journal of Future Computer and Communication*, *2*(5), 399–406. http://doi.org/10.7763/IJFCC.2013.V2.194

Ilyas, S. A. A. and M. (2012). *Near Field Communication Handbook*.

Indicadores, P., & Mercado, D. E. L. (2015). Boletín Técnico, 1–44.

Jin, X., & Wang, D. (2008). An Intelligent Model for Urban Demand-Responsive Transport System Control. *2008 International Symposium on Intelligent Information Technology Application Workshops*, 151–154. http://doi.org/10.1109/IITA.Workshops.2008.62

Jokinen, J. P., Sihvola, T., Hyyti??, E., & Sulonen, R. (2011). Why urban mass demand responsive transport? *2011 IEEE Forum on Integrated and Sustainable Transportation Systems, FISTS 2011*, 317–322. http://doi.org/10.1109/FISTS.2011.5973631

Kostic, B., & Gentile, G. (2015). Using Traffic Data of Various Types in the Estimation of Dynamic O-D Matrices, (June), 3–5.

Luo, X., Zhao, X., Sun, L., Ma, K., & Tang, J. (2014). Dynamic Bus Dispatching under the Environment of Internet of Things, 449–454.

Nunes, A. A., Galvao Dias, T., & Falcao e Cunha, J. (2015). Passenger Journey Destination Estimation From Automated Fare Collection System Data Using Spatial Validation. *IEEE Transactions on Intelligent Transportation Systems*, *17*(1), 133–142. http://doi.org/10.1109/TITS.2015.2464335

Pelletier, M. (2009). Smart Card Data in Public Transit Planning : A Review. *Transportation*.

Pérez, F., Velásquez, G., Fernandez, V., & Dorao, J. (2012). Movilidad Inteligente. *Minetur.Gob.Es*, 111–122. Retrieved from http://www.minetur.gob.es/Publicaciones/Publicacionesperiodicas/EconomiaIndustrial/RevistaEconomiaIndustrial/395/FIAMMA PEREZ y OTROS.pdf

Roberto, J., & Zamora-colín, U. (2013). Bus Rapid Transit (BRT) en ciudades de América Latina , los casos de Bogotá (Colombia) y Curitiba (Brasil) Bus Rapid Transit (Brt) in Latin America , the case of.

S.A., T. (2009). *Indice Anexo Técnico Descripción General del Sistema*.

S.A., T. (2016). *Respuesta Radicado 2016ER1912*.

Secretaría Transito y Transporte. (2006). Technical report: Transporte público, 1–138.

Standardization, I. O. for. (2000). Iso/Iec 14443-1, *5*(April).

Standardization, I. O. for. (2001a). Iso/Iec 14443-2.

Standardization, I. O. for. (2001b). Iso/Iec 14443-3, *2001*, 58.

Standardization, I. O. for. (2001c). Iso/Iec 14443-4, *2001*, 1,15.

Wang, W., Attanucci, J. P., & Wilson, N. H. M. (2011). Bus Passenger Origin-Destination Estimation and Related Analyses Using Automated Data Collection Systems. *Journal of Public Transportation*, *14*(4), 131–150. http://doi.org/10.5038/2375-0901.14.4.7

Wright, L. (2007). Bus Rapid Transit Planning Guide, (June), 836. http://doi.org/2007

BIBLIOGRAFICA EXTERNA

Clark, S. (2009). Telecom Italia and ATM to launch NFC ticketing service in Milan. NearFieldCommunicationsWorld.com.

http://pymbook.readthedocs.io/en/latest/flask.html