Diseño e implementación de una solución IoT para el sistema de Control de acceso en Cicloparqueadero inteligente

Trabajo de Grado

NICOLÁS PARRA S

JUAN PABLO MORENO C.

Director:

Angela Tatiana Zona Ortiz, PhD

Pedro Alejandro Mancera Lagos, MsC

UNIVERSIDAD SANTO TOMÁS
FACULTAD DE INGENIERÍA DE TELECOMUNICACIONES
BOGOTÁ, 2019

AGRADECIMIENTOS

Los autores agradecen a todos los miembros del proyecto Centro de Excelencia y Apropiación en Internet de las Cosas (CEA-IoT), especialmente al nodo USTA. Al igual que a todas las instituciones que financiaron este centro: Ministerio de Tecnologías de la Información y las Comunicaciones - MinTIC, y Departamento Administrativo de Ciencia, Tecnología e Innovación - Colciencias a través del Fondo Nacional de Financiamiento para la Ciencia, la Tecnología y la Innovación Francisco José de Caldas (ID del proyecto: FP44842-502-2015)

TABLA CONTENIDO

RESUMEN

El presente documento muestra el proceso de elaboración e implementación de un piloto a escala de laboratorio del sistema de control de acceso para un cicloparqueadero inteligente. El diseño se basa en una serie de requerimientos, que lleva a un sistema que está compuesto por una infraestructura física con varios Módulos Esclavos, un Módulo Contraseña y un Módulo Maestro; una interfaz web alojada en un bucket que sirve como medio de inscripción de usuarios al sistema de control de acceso; y una arquitectura en la nube basada en microservicios AWS y bases de datos no relacionales. Para el desarrollo de este proyecto se utilizò la metodología Scrum, y una vez finalizado el desarrollo, cada componente del sistema paso por una serie de pruebas para determinar su correcto funcionamiento.

Palabras clave: AWS, Microservicios, Modulo Escalvo, Módulo Contraseña, Modulo Maestro, Interfaz Web

ABSTRACT

The elaboration and implementation process of the access control system for an intelligent cycle park pilot at laboratory scale is showed. The system was designed based on a series of requirements, and is composed of a physical infrastructure, where the Slave Module, Password Module and Master Module will be found; a web interface stored in a bucket used to register users into the access control system; and a cloud architecture based on AWS microservices and non-relational databases.- To carry out this project, the Scrum methodology was used, then each system component was validated in order to verify its functionality .

Keywords: AWS; Microservices; Slave Module: Password Module; Master Module; Web Interface; Internet of Things ; Smart Campus ; Parking Access system

INTRODUCCIÓN

El uso de la bicicleta como medio de transporte va en aumento en las ciudades grandes [1], debido a que la movilidad por la ciudad se ha convertido en una tarea difícil, principalmente por la congestión en las avenidas principales, y porque el transporte público no cumple con la demanda de los usuarios que necesitan trasladarse de un punto a otro. Este aumento en el uso de la bicicleta ha convertido la bicicleta en un objetivo principal para los ladrones en las calles, pero a su vez también ha generado la posibilidad de creación de nuevos modelos de negocio que permitan la especialización de cicloparqueaderos que cumplan con los niveles de seguridad que los biciusuarios necesitan.

Bogotá tiene la infraestructura necesaria para la movilidad en bicicleta con 500km de ciclorutas [2], de esta manera los ciudadanos tienen la posibilidad de usar la bicicleta como medio de transporte.

En el presente proyecto, se presenta un diseño y una prueba a nivel de laboratorio de un sistema de control de acceso para el cicloparqueadero de la sede principal de la Universidad Santo Tomás en la ciudad de Bogotá. Este sistema de control de acceso implementa sistemas de hardware, una interfaz web y una arquitectura en la nube basada en microservicios. El sistema cuenta con servicios de valor agregado como: la generación de notificaciones al usuario vía Email del espacio de parqueo en el que quedó guardada su bicicleta, y la visualización de reportes estadísticos sobre el comportamiento de todo el sistema por parte de un administrador a través de la interfaz web.

El sistema propuesto en este trabajo se puede integrar con los otros sistemas del cicloparqueadero inteligente de la Universidad Santo Tomás (ParkUrBike), como el sistema de disponibilidad y el sistema de monitoreo ambiental. El sistema de control de acceso se diseña y desarrolla bajo los estándares de Sistemas Inteligentes de Transporte (SIT) [3] en el marco de ciudades inteligentes, con miras de hacer un producto sólido de cicloparqueadero inteligente que permita la escalabilidad a varios parqueaderos y empresas, de manera que los datos generados permitan la toma de decisiones en una ciudad u organización.

En el capítulo 1 se realiza una contextualización sobre Internet de las Cosas, los modelos de conectividad existentes y la arquitectura de una solución IoT planteada por Gartner [1]. En el capítulo 1 se habla sobre el marco general del proyecto. El capítulo 2 se hace una contextualización sobre ciudades inteligentes enfocado en el estándar de Sistemas Inteligentes de Transportes (SIT) y se define qué es ParkUrBike. El capítulo 3 se encuentran los requerimientos para el sistema de control de acceso para ParkUrBike. El capítulo 4 se habla sobre el diseño, desarrollo y validación de la infraestructura física. El capítulo 5 trata el diseño, desarrollo y validación de la plataforma en la nube del proyecto. Finalmente, las conclusiones son presentadas en el capítulo 6.

1 MARCO GENERAL DEL PROYECTO

En el presente capítulo se muestra de forma general el proyecto “Diseño e implementación de una solución IoT para el sistema Control de acceso en Cicloparqueadero inteligente”, abordando el objetivo general, los objetivos específicos, el alcance y la metodología del mismo.

1.1 OBJETIVOS

Diseñar y validar un sistema de control de acceso para un cicloparqueadero inteligente basado en tecnología IoT, que permita la asignación de un espacio de parqueo vacío a un usuario registrado en el sistema, brindando algún nivel de seguridad en el ingreso y retiro de la bicicleta.

- Establecer un mecanismo para registro de usuarios en el sistema y su bicicleta, y la visualización de indicadores de uso para un administrador del biciparqueadero en una interfaz web.
- Diseñar e implementar un bus serial de cerraduras inteligentes que se abran de manera selectiva según una contraseña asignada a un usuario, ya sea para ingreso o retiro de su bicicleta.
- Desarrollar la inteligencia del sistema de control de acceso basada en microservicios para permitir la identificación de un usuario registrado, la generación de la contraseña para ingreso y retiro de la bicicleta, y la asignación de un espacio vacío para el ingreso.
- Validar el funcionamiento del sistema de control de acceso mediante la implementación de un piloto de 4 cerraduras.

1.2 ALCANCE

Un piloto a escala de laboratorio del sistema de control de acceso para un cicloparqueadero inteligente compuesto por: 4 cerraduras conectadas en bus, una interfaz web, una inteligencia basada en microservicios AWS y bases de datos no relacionales.

1.3 METODOLOGÍA

La figura 1 muestra la metodología de 3 fases, utilizada en el proyecto. Estas fases son: Diseño, Desarrollo y Validación.

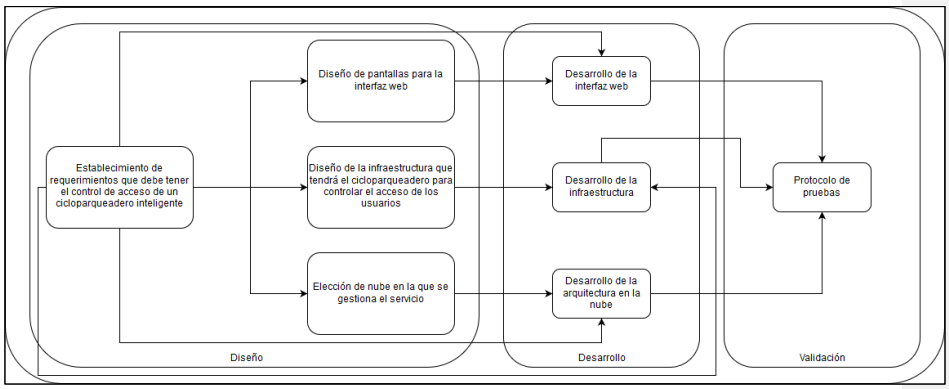


Figura 1. Metodología del proyecto

Fuente: Autores.

En la fase de Diseño se definen los requerimientos generales de la solución IoT, se realiza de diseño de cada uno de los componentes del sistema, de manera que al final de la fase se cuenta con el diseño de pantallas de la interfaz web, diseño de la infraestructura física y la elección de la nube.

En la fase de Desarrollo se realiza el desarrollo de cada uno de los componentes del sistema, siguiendo los diseños y requerimientos de la fase de Diseño.

La fase de Validación se realiza el proceso de validación por medio de varios protocolos de pruebas, que permite la verificación del correcto funcionamiento de cada componente y luego de la integración de estos para conformar el sistema, validando el cumplimiento a cabalidad de los requerimientos estipulados en la fase de Diseño.

El marco metodológico del proyecto es SCRUM [4]. SCRUM es un método utilizado para el trabajo en equipo, que se basa en abordar de manera cíclica cada una de las tareas de un

listado hasta cumplir el objetivo, cada ciclo es llamado iteraciones o Sprint, y suele planificarse por semana (Figura 2).

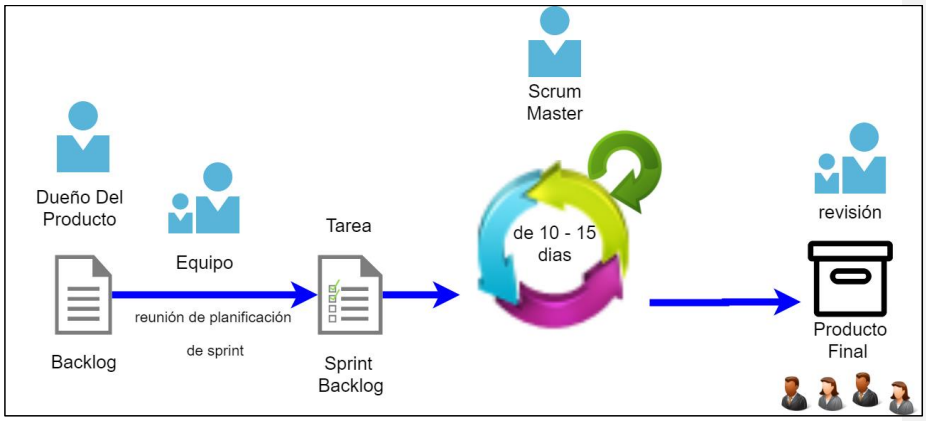


Figura 2. Marco metodológico SCRUM

Fuente: Autores.

Para este proyecto los sprints fueron semanales, cada semana se realimenta el desarrollo del proyecto, realizando mejoras y/o cambios que permitan adecuar la implementación en curso. La finalidad de estas mejoras o cambios es permitir mayor flexibilidad en la ejecución de los procesos de ingreso y retiro de una bicicleta, y registro de nuevos usuarios al sistema.

2 INTERNET DE LAS COSAS Y CASOS DE USO

Si bien actualmente no existe una definición puntual y única sobre qué es Internet de las cosas (IoT[1]), se entiende de manera general como la conexión de objetos a Internet o una red para la generación de datos. Haciendo referencia a escenarios donde sensores y artículos de uso diario, que habitualmente no se consideran equipos de cómputo, generan valor mediante el intercambio de información entre ellos [5].

La reciente inclinación por el uso de tecnologías autónomas dentro de la oferta y productos del mercado en general, ha permitido que la potencialización e impulso de Internet de las cosas sea cada vez más una realidad [6].

Hablar de IoT también incluye conectividad omnipresente, adoptando protocolos de transporte como IP, la economía en la capacidad de cómputo, la miniaturización, lo avances en el análisis de datos y el surgimiento de la computación en la nube.

2.1 MODELOS DE CONECTIVIDAD PARA SOLUCIONES IOT

La implementación de soluciones IoT se basa en diferentes modelos de conectividad, cada uno de los cuales tiene sus propias características en el transporte de información.

Los cuatro modelos de conectividad para arquitecturas IoT incluyen:

1. Dispositivo a dispositivo (*Device-to-Device*)
2. Dispositivo a nube (*Device-to-cloud*)
3. Dispositivo a puerta de enlace (*Device-to-Gateway*)
4. Intercambio de datos a través del back-end (*Back-end data sharing*)

Estos modelos destacan la flexibilidad en las formas en que los dispositivos IoT pueden conectarse y proporcionar valor para los usuarios.

[1] Por sus siglas en inglés, Internet of Things (IoT)

2.1.1 Device-to-Device

La conectividad *Device-to-Device* involucra la comunicación entre 2 o más dispositivos que no requieran la intervención humana, estas tecnologías buscan implementar nuevos o diferentes escenarios de mercado, basados en esfuerzos a bajo costo de implementación, potencialmente escalables con capacidad de un mayor número dispositivos conectados y un tráfico de datos muy bajo.

Dentro de este modelo, dos dispositivos finales tienen la capacidad de conectarse de forma inmediata sin la necesidad de que exista algún tipo de comunicación intermediaria (Figura 3). Estos dispositivos se pueden conectar por medio de protocolos Local/Personal/Wide Area Network (LAN/PAN/WAN) [7]. Los dispositivos se comunican por datagramas IP, los dispositivos conectados allí tienen la capacidad de tomar decisiones realizando tareas específicas, como encender un motor, disparar una alarma, encender luces, etc. La red de conmutación de paquetes solo se encargan de transportar el tráfico para que lleguen los datagramas hacia su destino de forma correcta [7].

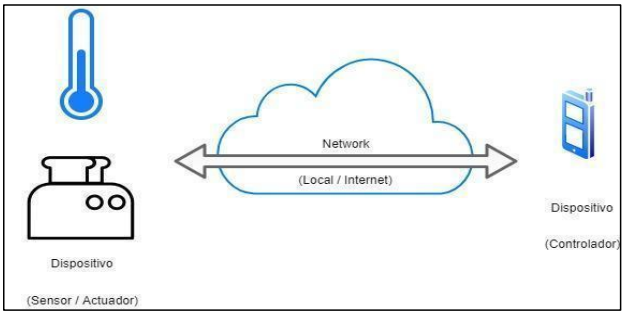


Figura 3. Mensajería MQTT

Fuente: autor

2.1.2 Device-to-Cloud

La mayoría de las soluciones IoT, utilizan protocolos de transporte UDP o TCP dependiendo del servicio que se vaya a implementar. UDP se utiliza para aplicaciones no críticas, las cuales no son sensibles a la pérdida de paquetes o al orden de llegada de los mismos. Mientras que TCP se orienta a aplicaciones sensibles a requerimientos de pérdida de paquetes. Por otra parte, existen protocolos de mensajería a servidor que son muy comunes

en soluciones IoT, de manera estándar son protocolos de la capa de sesión para el envío de datos desde un dispositivo a un servidor en la nube; algunos de estos protocolos para implementaciones de servicios IoT son:

- MQTT[2]: es un protocolo diseñado para proporcionar conectividad entre aplicaciones con redes de comunicaciones. Sigue una arquitectura de publicación/suscripción, donde hay 3 componentes principales: publicadores, suscriptores y un intermediario (*broker*) (Figura 4). En una solución IoT, los publicadores son sensores o actuadores que envían al broker datos bajo un *topic*, los suscriptores son las aplicaciones interesadas en recibir estos datos, para tal fin se suscriben a los *topics* publicados por el broker. Y los brokers son un punto de encuentro para los publicadores y los suscriptores, a través del uso de *topics*.
- AMQP[3]: es un protocolo de inicio de sesión que se ejecuta sobre la capa de transporte y corre sobre TCP, este protocolo tiene una arquitectura similar a MQTT con la diferencia de que el *broker* se divide en dos componentes principales: el intercambio y las colas (Figura 5). La información transportada debe ser puesta en colas que son previamente son configuradas y procesadas dependiendo de su relevancia. Básicamente, las colas presentan los *topics* y los suscriptores obtendrán los datos de los sensores siempre que se encuentren disponibles en la cola.

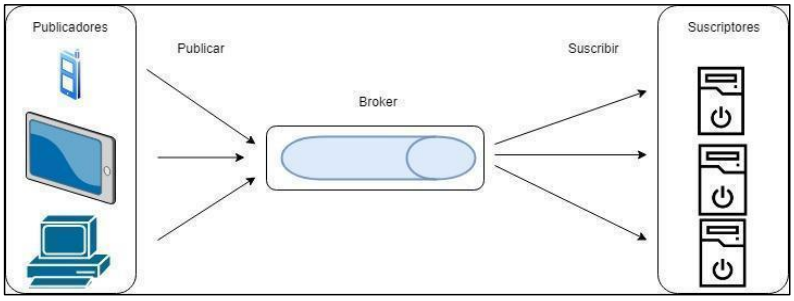


Figura 4. Arquitectura MQTT

Fuente: Autores.

[2]Del inglés, *Message Queue Telemetry Transport.*

[3] Del inglés, *Advanced Message Queuing Protocol*

➢

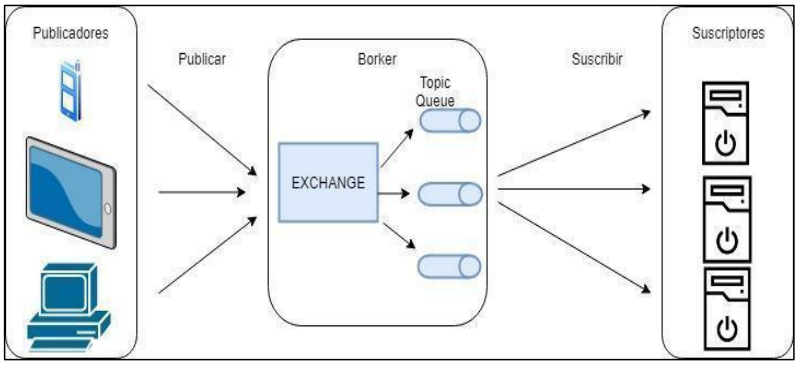


Figura 5. Arquitectura AMQP

Fuente: Autores.

- CoAP[4]: es un protocolo de la capa de sesión la cual está diseñada para un entorno de trabajo RESTful es decir, bajo interfaces HTTP y comunicación directa con servidores. Sin embargo, debido al tamaño de las tramas, el uso de este protocolo sería una carga grande y un consumo de energía bastante alto, para aplicaciones livianas de soluciones IoT. Este protocolo está diseñado para permitir que datos medidos, censados, monitoreados y capturados por dispositivos de baja potencia tengan la capacidad de ser analizados desde un servidor permitiendo comunicación casi directa. Este protocolo funciona sobre UDP, en lugar de hacerlo por TCP como tradicionalmente se hacía en HTTP teniendo un mecanismo ligero para proporcionar confiabilidad [8].

  La arquitectura que proporciona este protocolo se encuentra dividida en 2 capas, la capa de mensajería, y la capa de solicitud y respuesta. La capa de mensajería es responsable de la confiabilidad y la duplicación de los mensajes, mientras que la otra capa se encarga únicamente de la comunicación entre dispositivo y servidor, existen dos tipos de modos de mensajería: confirmable y no confirmable. En el modo confirmable al momento de realizar una solicitud al servidor, este responde de

[4] Del inglés, *Constrained Application Protocol*

manera inmediata, es decir, con un acuse de recibo o ACK[5]. Mientras que en el modo no confirmable, al realizar una solicitud al servidor, este no responde con un acuse de recibido o ACK. Al igual que el protocolo HTTP, el CoAP también utiliza solicitudes de tipo GET, POST, PUT, PUSH, DELETE [9].

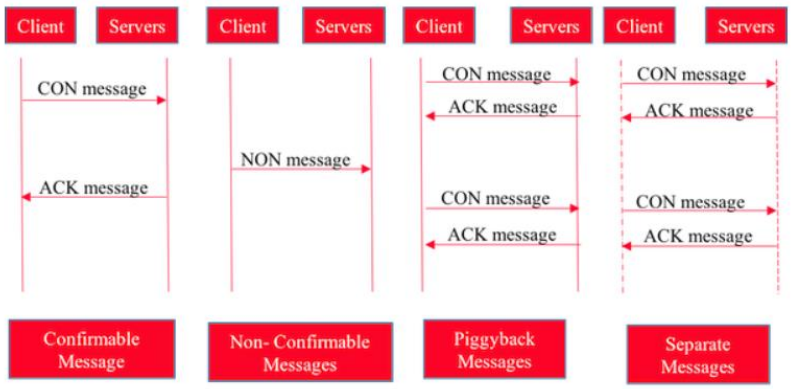


Figura 6. Mensajería CoAP

Fuente: RFC7252.

2.1.3 Device-to-Gateway

En el modelo de dispositivo a puerta de enlace ALG[6], el dispositivo IoT se conecta a un servicio en la nube o un application server a través de ALG como un canal directo. Esto es muy útil cuando se dispone de dispositivos que no manejan protocolos como HTTP o CoAP y su comunicación es por medio alámbricos como puertos seriales o inalámbricos como wifi, bluetooth o zigbee. En términos generales, esto significa que actúa como un intermediario entre los dispositivos del sistema y los servicios en la nube, proporcionando control y seguridad al tráfico de información del sistema (Figura 6).

Se espera que, en un futuro, existan gateways de bajo costo y con una infraestructura mucho menos compleja para clientes, usuarios finales, empresas y entornos industriales.

[5] Del inglés, Acknowledgement

[6] Del inglés Application Layer Gateway

Como así mismo existan dispositivos IoT que tengan la capacidad de realizar solicitudes más complejas o tengan la capacidad de conectarse a internet y no requiera ALG que transmitan los datos desde el dispositivo hacia la nube.

La evolución de los sistemas usando un modelo de comunicación device to gateway aún se encuentra en desarrollo y su principal característica es la capacidad de realizar interoperabilidad entre dispositivos que manejen diferente lenguaje [10].

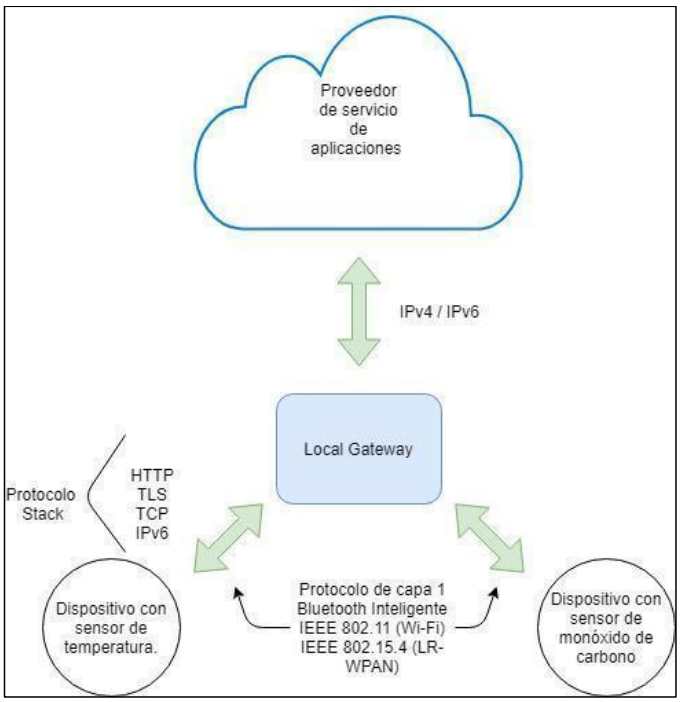


Figura 6. Consideraciones en arquitectura para redes de objetos inteligentes.

Fuente: Autores.

2.1.4 Back-end data Sharing

El modelo de compartir información por back-end, se refiere a una arquitectura de comunicación que habilita al usuario a exportar y analizar objetos de datos de forma

inteligente desde un servicio en la nube en combinación de otras diferentes fuentes de información.

En términos generales, una arquitectura de intercambio de datos a través del back-end es aquella que recolecta datos desde uno o varios dispositivos IoT para ser almacenados y analizados posteriormente.

El modelo back-end [11] de intercambio de datos sugiere que se necesita un enfoque de servicios en la nube federados, es decir máquinas virtuales, almacenamiento de archivos, bases de datos administradas o interfaces de acceso a aplicaciones (API) para lograr la interoperabilidad de los datos de dispositivos inteligentes alojados en la nube (Figura 7).

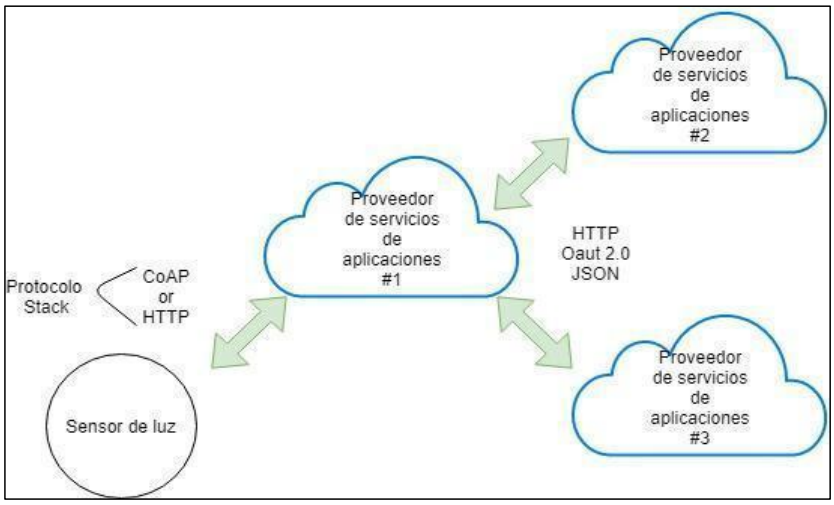


Figura 7.Consideraciones en arquitectura para redes de dispositivos inteligentes en redes IoT.
Fuente: Autores.

2.2 ARQUITECTURA GENERAL IOT

Gartner propone una arquitectura para el desarrollo de soluciones IoT, contemplando cualquier grado de complejidad para su implementación (Figura 8).

De manera general, esta arquitectura se divide en 4 capas que están conectadas de manera jerárquica, iniciando por los componentes físicos hasta el manejo de los datos recopilados que dan valor al negocio. En la arquitectura se contempla que la captura de datos se obtiene desde las capas inferiores donde se encuentran los componentes físicos, mientras que el control es suministrado desde las capas superiores.

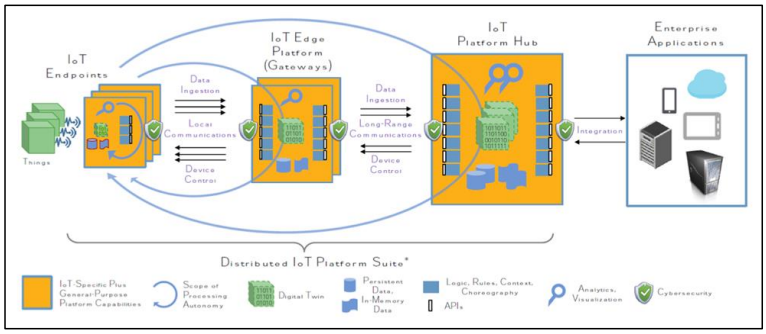


Figura 8. Modelo de referencia para una solución de negocio IoT

Fuente: Use the IoT Platform Reference Model to Plan Your IoT Business Solutions, Gartner.

2.2.1 IoT Endpoints Layer

En esta capa se encuentran situados los objetos que miden, sensan, monitorean, detectan y capturan eventos en instantes específicos de manera sincrónica o asincrónica. Según las recomendaciones de Gartner estos dispositivos transmiten los datos capturados con un procesamiento mínimo, descartando posibles datos basura que se puedan generar durante el funcionamiento del sistema, es decir que capturan y envían los datos relevantes hacia un Gateway ubicado en la red.

La transmisión de datos que se generan en esta capa pueden ser por un medio inalámbrico o alámbrico. Generalmente los medios inalámbricos son utilizados, reduciendo la posibilidad de que se presenten daños en el medio de comunicación (ruptura línea de transmisión, accidentes en la instalación del servicio y reduciendo costos para el despliegue de la solución). Sin embargo, el medio inalámbrico está expuesto a interferencias por parte de objetos externos que emitan ondas de radio a la frecuencia de operación, y el alcance

se encuentra limitado por las potencias que pueda emitir tanto el dispositivo como la del equipo al que se estén conectando.

2.2.2 IoT Edge Platform

En esta capa se encuentran dispositivos más “inteligentes” que tienen la capacidad de conectarse a los demás objetos y recopilar esa información para realizar procesamiento y el transporte de los mismos. Un dispositivo de estos tiene la capacidad de atender a todos los dispositivos de la primera capa, dependiendo de la topología, la capacidad de procesamiento, la memoria física, los protocolos de transporte y el tamaño de las tramas que circulan por el sistema.

Estos dispositivos tienen también la capacidad de conectarse a la capa superior (IoT Platform) mediante interfaces de aplicación (API[7]), protocolos de hipertexto como HTTP o protocolos de mensajería como MQTT.

Las grandes compañías de nube como AWS, Microsoft Azure y Google Cloud desarrollaron sus propias soluciones para asegurar que los usuarios que desarrollen soluciones IoT utilicen su plataforma en la nube. Ofreciendo PaaS[8], un servicio que permite alquilar recursos de cómputo en la nube pagando únicamente por el uso que se hace de ellos, e integrarlo con la solución IoT que esté diseñando el usuario y tenga conexión directa con los demás servicios en la nube.

AWS anunció en el Re:invent del 2016 [12] el lanzamiento de Greengrass, un producto que es integrable con cualquier solución IoT, permitiendo a una solución IoT acceder a los servicios de AWS de forma local sin necesidad de una conexión constante internet. Azure con IoT Edge busca cautivar al usuario desde el principio, no solo ofreciendo PaaS si no también SaaS[9] desarrollando para el usuario final integrarse de manera directa el dispositivo con los servicios en la nube y ofreciendo una interfaz interactiva en la que puede gestionar su infraestructura [13]. Por su parte, Google Cloud con Edge TPU busca integrar

[7] Del inglés, Application Programming Interface

[8] Del inglés Platform as a Service

[9] Del inglés Software as a Service

inteligencia a esta capa, desplegando la posibilidad de hacer inferencias de aprendizaje a soluciones o servicios IoT, que sean replicables en cualquier sitio con la misma unidad de aprendizaje, centrando toda la información almacenada y procesada en una misma máquina virtual que ellos proveen [14].

2.2.3 IoT Platform

Esta capa se encarga del procesamiento, manipulación, análisis y almacenamiento de los datos que son transportados por las capas anteriores. Se podría decir que es un repositorio donde debe llegar todos los datos o toda la información; por lo general se utilizan plataformas comerciales de computación en la nube [15] (o en inglés Cloud Computing) como AWS (Amazon Web Services), Microsoft Azure, IBM Hadoop, Ubidots, entre otras. Fruto del análisis de los datos recopilados, posteriormente es posible generar notificaciones o incluso alertas tempranas, evitando, en el peor de los casos, catástrofes o simplemente permitiendo realizar mantenimientos preventivos de equipos [16].

Es muy común ver la integración de soluciones IoT con microservicios,prestados por las plataformas de Cloud Computing [15], que permiten la manipulación de datos sin necesidad de tener un servidor. Solo se ejecuta un código que tiene una entrada, un procesamiento y una salida que puede ser el resultado del análisis de la información entrante; minimizando costos operativos y evitando la necesidad de instalar servidores locales para el manejo de los datos.

2.2.4 Enterprise Applications

Esta capa es la más importante, es la que le da valor a toda la información que fue recopilada, almacenada y procesada en las capas anteriores, es donde se marcan directrices o se ejecutan procesos. Es aquí en donde la información recopilada se convierte en servicios de valor agregado para los usuarios finales, y se evidencia la optimización en costos de capital y operativos al implementar soluciones IoT [17].

2.3 HERRAMIENTAS DE DESARROLLO

Para el desarrollo de una solución IoT es importante saber que los datos generados por los dispositivos que se conectan a internet deben ser administrados, procesados y almacenados para generar valor al servicio, o la posibilidad de tomar decisiones gracias a

estos datos. Es por ello que para desarrollo de soluciones IoT existen las herramientas mencionadas a continuación.

2.3.1 Enterprise Applications

El crecimiento de la tecnología ha hecho que los dispositivos IoT evolucionen con diferentes protocolos de comunicación, lo que dificulta en cierta manera la integración con redes tradicionales que ya poseen datos estructurados. También se presenta un gran desafío al intentar realizar compatibilidad con los protocolos *legacy* o sin actualización. Típicamente, los dispositivos IoT se han diseñado para que el consumo de energía sea mínimo y, como tal, por tal razón los recursos son limitados (almacenamiento, procesamiento, entre otros). Pero también se han convertido en un desafío desde la perspectiva de un dispositivo IoT debido a que se espera tengan una larga vida útil con un costo mínimo, mientras procesan, sensan, capturan datos, y consumiendo servicios.

Una arquitectura de microservicios, presenta la posibilidad de facilitar el desarrollo de una solución IoT. Ya que, promueve la implementación de servicios independientes, la atomicidad del servicio, la solución a fallas en puntos únicos, la administración de transacciones de seguridad en el tráfico de la red y la sincronización de la comunicación; gracias al manejo de API's hacía microservicios que ofrecen realizar tareas específicas a un costo mínimo y con respuestas casi inmediatas. Una arquitectura de identificación y clasificación basada en microservicios para una solución IoT se divide en servicios como [18]:

- **Funcionales:** son servicios que soportan operaciones funcionales para un sistema inteligente dentro del dominio de un servicio IoT. En IoT, por lo general cumplen funciones muy específicas (almacenar, contabilizar, identificar, sumar, entre otros) que en su mayoría son servicios utilizados por sistemas externos o dispositivos inteligentes.
- **No funcionales:** son servicios que están diseñados para realizar tareas que no sean operacionales (como autenticación, monitoreo, autorización, entre otros) pero son utilizados para operaciones de confianza del sistema. Estos servicios no están expuestos para que puedan ser utilizados por sistemas externos, ni por dispositivos inteligentes que busquen integrarse a una solución IoT.

Los servicios funcionales y no funcionales están claramente identificados y en una arquitectura IoT (Figura 9), la manera en que se puede acceder a este tipo de servicios es a través de una invocación a un API, que recibe la solicitud y dependiendo del contenido que tenga el mensaje se activan los servicios no funcionales o podría hacer que desde los servicios funcionales se oculten los servicios no funcionales ante cualquier otro sistema.

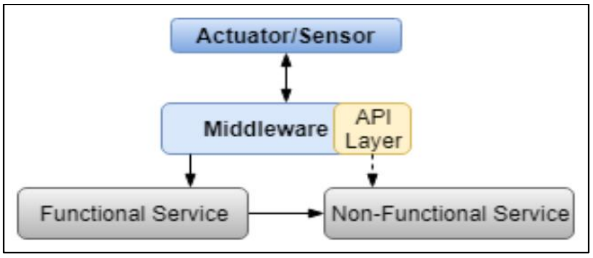


Figura 9. Microservicio para la identificación y clasificación del servicio IoT
Fuente: IoT Architectural Framework: Connection and Integration Framework for IoT Systems, Onoriode Uviase & Gerald Kotonya.

La figura 9 es un ejemplo de una arquitectura basada en microservicios de un sistema sensor/actuador, en la cual al ocurrir un evento en específico se invoca un API realizando requerimientos específicos a servicios funcionales o no funcionales dependiendo del caso que se presente en dicho evento [18].

2.3.2 Bases de datos

Una base de datos está definida como una colección de datos estructurados. Tradicionalmente se implementan las conocidas bases de datos relacionales que están definidas bajo un lenguaje de consulta estructurado SQL[10]. Actualmente la tendencia en el mercado es la implementación de las bases de datos no relacionales conocidas como NoSQL [19]. Esta tendencia se debe a que el nuevo modelo de base de datos en

[10] Del inglés structured query language

comparación ofrece mejores resultados al aplicar el teorema de CAP[11] que explica la consistencia, la disponibilidad y la tolerancia a participación, a saber:

- Consistencia: Se refiere a que una vez se finaliza la operación de actualización, todos pueden leer la última versión de los datos almacenados en las bases de datos. Un sistema que funcione de esa manera se considera un sistema consistente.
- Disponibilidad: Se refiere ofrece un funcionamiento continuo que sea tolerante a fallos. En la implementación de una base de datos se haría como un grupo de nodos, de tal forma que, en cualquier fallo de un nodo, los demás puedan seguir realizando consultas o escribiendo sin que todo el sistema colapse.
- Tolerancia a partición: Se refiere a que, si un nodo de los que compone una base de datos se llega a caer, a fallar o a ser inaccesible, tenga la facilidad de seguir funcionando redirigiendo las consultas a algún nodo activo del sistema.

2.3.2.1 Bases de datos relacionales

Las bases de datos tradicionales como las SQL, se enfoca en la consistencia y admiten las propiedades ACID[12], una serie de parámetros que permiten clasificar bases de datos. De manera general se refiere a:

- Atomicidad se refiere a la manera en que se descartan las transacciones exitosas o no
- Coherencia se refiere a la acción que toma el sistema cuando un caso de transacción fallida se presenta, que, por lo general, vuelve al estado anterior antes de que ocurriera el fallo y vuelve a realizar la transacción, por lo tanto, el sistema permanece estable.
- Aislamiento cuando las transacciones se completan de forma exitosa sin ningún tipo de fallo y se procesan de forma independiente.
- Durabilidad que se refiere a cuando una transacción se confirma como exitosa, es almacenada en los registros y no se perderán, permitiendo que el sistema se pueda recuperar en caso de alguna emergencia.

[11] Acrónimo de Consistency, Availability and Partition Tolerance

[12] Acrónimo de Atomicity, Consistency, Isolation and Durability

2.3.2.2 Bases de datos No relacionales

Por otro lado, las NoSQL se centran más en la disponibilidad y partición dando consistencia eventual, y al igual que las SQL, siguen ciertas características de las propiedades BASE.

BASE no es estrictamente un sistema con propiedades consistente. En tales sistemas, los clientes pueden encontrar inconsistencias en los datos cuando se presentan procesos de replicación o actualización en curso. Pero al finalizar los datos estarán correctamente almacenados bajo la misma estructura; en aplicaciones de soluciones IoT, el número de usuarios, los requisitos de velocidad y demandas que aumentarán constantemente, por lo tanto, la tolerancia a particiones debe ser algo que se tiene que considerar desde un principio. Las NoSQL da alta prioridad a la disponibilidad sobre la consistencia, por lo que es más fácil replicar este modelo que el tradicional SQL que sigue en orden inverso.

2.3.2.3 Relacionales vs No Relacionales para una solución IoT

Cómo anteriormente se mencionó las bases de datos SQL se basan en el modelo de datos relacionales para almacenar los datos, en este modelo se almacenan en filas y columnas de forma tabular, donde estas tablas pueden estar relacionadas entre sí implementando métodos como las llaves primarias o las llaves foráneas.

Las bases de datos NoSQL siguen el mismo modelo que las SQL. El modelo no relacional admite el almacenamiento de datos sin esquemas en diversas formas. Estas pueden crearse sin diseñar un modelo de tablas inicialmente, es decir, son adaptables a cualquier estructura de datos en una sola tabla. Con características como la escalabilidad horizontal, almacenamiento sin esquema, donde la estructura NoSQL se vuelve competente para almacenar datos de dispositivos IoT [20]. La popularidad de estas bases de datos está relacionada con las características que proporciona, entre ellas, la escalabilidad y la posibilidad de tener una arquitectura distribuida.

Elegir la base de datos que va a soportar una solución IoT se vuelve una tarea obligatoria a la hora de empezar a realizar el diseño del sistema, desde el punto de vista de una arquitectura IoT, las diferencias entre los dos tipos de bases de datos actualmente existentes son:

- Escalabilidad: SQL ofrece escalabilidad vertical, mientras que NoSQL ofrece escalabilidad horizontal. Escalar verticalmente implica que los recursos a nivel de cómputo sean dedicados a un mismo nodo, lo que implica a futuro un mayor costo en el desempeño del sistema, mientras que la escalabilidad horizontal el número de servidores o nodos se incrementa solo para compartir la carga de los datos del sistema, lo cual ayuda en la implementación de una solución IoT ya que no se necesita invertir mucho en almacenamiento inicialmente y se incrementaría la capacidad de las bases de datos conforme a la demanda.
- Recuperación de datos: En SQL las tablas al estar relacionadas entre sí, para realizar una búsqueda es necesario hacer un JOIN para visualizar los datos que hay en la relación que hay entre dos tablas que requería más capacidad de procesamiento para las búsquedas y consumían mucho tiempo. Por otro lado, las NoSQL almacenan objetos que pueden contener arreglos de objetos dentro de cada columna o inclusive más objetos, eliminando la necesidad de combinar vistas para realizar búsquedas, mejorando capacidad de procesamiento y tiempos de búsqueda más cortos.
- Madurez del sistema: Las SQL al llevar tanto tiempo en el mercado, se han estandarizado y han mejorado mucho en temas de seguridad como la autenticación, la integridad de sus tablas o la confidencialidad; mientras que en las NoSQL al estar tan poco tiempo en el mercado, aún se ven falencias en la seguridad de los datos, para esto se han venido implementando herramientas intermedias que sí posean protocolos de seguridad de la información de los usuarios y en las cuales, la herramienta haga las solicitudes directamente a la base de datos de forma interna [20].

2.4 CASOS DE USO

2.4.1 Seguridad inteligente basada en IoT para una organización

En [6] se describe una solución IoT de seguridad flexible y de bajo costo, cuyo propósito es brindar mayor seguridad para mantener la información de una organización de manera adecuada a un costo razonable. Este sistema tiene dos módulos principales: el módulo de hardware (compuesto por Arduino, GSM, Bluetooth y sensores), y el módulo de comunicación de software (aplicación web y aplicación de Android). Todas las

comunicaciones del módulo de hardware pasan por el microcontrolador de Arduino maestro el cual, para este sistema, sirve como un micro-servidor web y también como gateway para todos los módulos de hardware. Este desarrollo se dividió en dos partes, la primera se enfocó en el control y organización de los dispositivos, y la segunda en controlar la seguridad de la organización.

La arquitectura implementada para esta solución se basa en un servicio de monitoreo de sensores y actuadores. La información capturada por estos dispositivos son enviadas a un repositorio aplicando protocolos de seguridad, donde allí se procesan y se disparan alertas tempranas en caso de fallas. La idea se basa en que exista un servicio web o una aplicación móvil que permite estár todo el tiempo informado del estado actual de la organización y poder, de manera remota, realizar acciones que cambien su funcionamiento actual. Así mismo que el sistema tenga la capacidad de realizar alertas a entidades públicas en casos de emergencia, como estaciones de policía o estaciones de bomberos (Figura 10).

El sistema propuesto ofrece dos características principales, el control de ingreso para la detección de intrusos y el monitoreo constante de humo y temperatura para la detección de incendios [6]

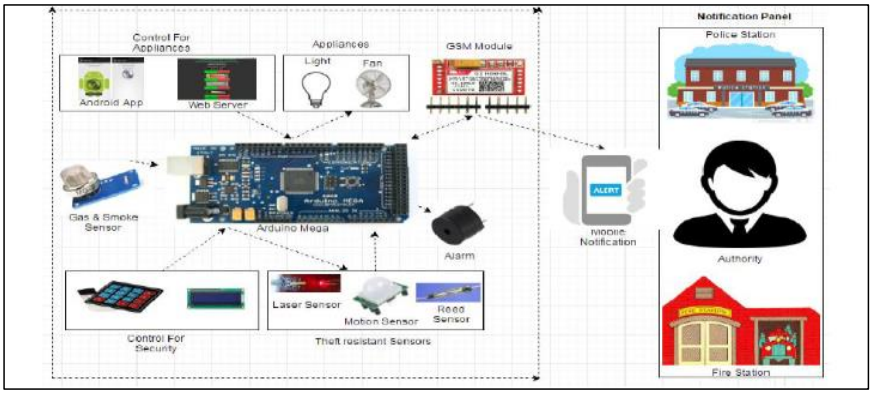


Figura 10. Arquitectura del sistema de alertas tempranas

Fuente: IoT Smart Security for an Organization based on IoT

2.4.2 Autenticación y control de acceso en IoT

En IoT, la comunicación debe establecerse de manera segura entre varios dispositivos; cuando a un dispositivo recibe información de otro, el mensaje recibido o información recibida debe ser consistente con la solicitud realizada inicialmente, es decir, que ambos dispositivos deben hablar el mismo lenguaje.

Por lo general, un protocolo de autenticación tiene varias tareas, entre ellas, el establecimiento, cambio o consulta de contraseñas de identificación. Este proceso es básicamente como un remitente puede adquirir una llave de sesión a través de la identificación del mensaje en la solicitud. En el protocolo de establecimiento de claves autenticadas, las llaves establecidas para establecer un canal dedicado a un dispositivo son igual de importantes que los datos que están siendo transportados.

Este proyecto propone una arquitectura IoT para soluciones de propósito general. Dicha arquitectura está basada en arquitecturas estándar para soluciones IoT (Figura 11).

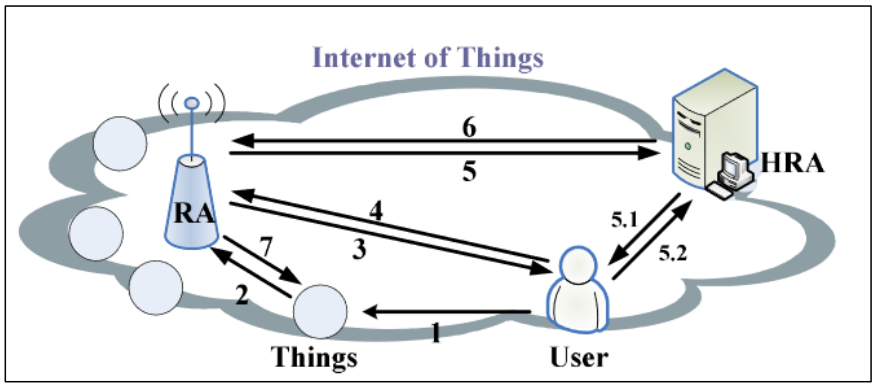


Figura 11. Ejemplo de arquitectura IoT

Fuente: Authentication and Access Control in the Internet of Things

La manera en que un dispositivo IoT se autentica frente a un servicio es diferente a la manera en que se autenticaría un usuario, ya que un dispositivo IoT se encuentra en un dominio diferente. sin embargo, en ambos casos se interactúa con el sistema, la diferencia

radica en que la interacción con el sistema se hace en diferentes niveles de la red. Entendiéndose diferente dominio como la interacción con un dispositivo maestro, en el que a un usuario se le asigna una identificación única mientras se encuentre usando el sistema y a un dispositivo se le asignan llaves de sesión que lo habilitan a intercambiar información el sistema.

Implementar un método de autenticación central implicaría que el diseño de la solución tenga en cuenta un centro de distribución de llaves (en inglés Key Distribution Center), pero esto sería un gasto de energía alto en el desarrollo del servicio. Razón por la cual en [21] se plantea un proceso de autenticación para una solución IoT de 7 pasos.

1. Que el usuario pueda acceder al dispositivo.
2. El dispositivo envía una solicitud de autenticación a su RA (Registration Authority) con fines de verificación.
3. Solicitud de ID de usuario de RA.
4. Respuesta del usuario con información HRA (Home Registration Authority).
5. RA verifica la información HRA del usuario y envía la solicitud de verificación de ID a la HRA.
   5.1 Desafía al usuario con una pregunta de seguridad.
   5.2 Respuesta del usuario ante la pregunta.
6. ID de la respuesta HRA OK o no.
7. RA responde al dispositivo sobre el ID del usuario y que emita una clave de sesión al usuario que acabo de ingresar al sistema.

2.4.3 WAZE

WAZE es un servicio que permite en tiempo real calcular rutas de un punto a otro, reportar el estado del tráfico, y realizar cambios de rutas de forma automática para que el usuario pueda llegar más rápido a su destino.

La información que utiliza WAZE para prestar estas funcionalidades es proporcionada por el usuario, más específicamente por el dispositivo móvil del usuario que informa al sistema la velocidad a la que mueve el usuario y otros datos ingresados por los mismos usuarios

como eventos específicos de accidentes de tránsito, robos en curso, trancones en la vía, estados de la vía, entre otros.

El futuro de los automóviles será enfocado hacía el internet de las cosas ya que se tiene la posibilidad de generar datos propios sobre el estado actual de los componentes internos del vehículo, sensar datos externos, identificar que el usuario se encuentra ubicado en un trancón por la velocidad del vehículo, enviar alertas en caso de accidente o robo, entre otras aplicaciones [22].

Google adquiere WAZE en el año 2013 por cerca de 1.000 millones de dólares americanos [23], con el fin mejorar Google Maps, pero con la llegada de Google Auto el enfoque cambió. Google Auto es una pantalla integrable en los automóviles que está diseñada para interactuar con el conductor, con la capacidad de conectarse a través de un smatphone con Android para poder acceder a las aplicaciones desde allí.

Google Auto llega justo para integrarse con el modelo que utiliza WAZE, crea rutas para sus usuarios por medio de datos recopilados históricamente por los usuarios de la aplicación en tiempo real. El futuro que se plantea para la movilidad de los vehículos es integrarlos con WAZE a través de Google Auto, donde los datos son capturados para realizar el cálculo de rutas [24].

2.4.4 Sensores inteligentes de agua para controlar la calidad de agua en los ríos

La empresa Libelium lanzó al mercado en el año 2014 un producto llamado *Smart Water*, que es una red de sensores inalámbricos a los que se les puede hacer monitoreo de forma remota. Este producto sensa al menos una docena de parámetros para verificar la calidad del agua. Además, es la primera plataforma de detección de la calidad del agua que cuenta con nodos autónomos, que se conectan a la nube para el control de la calidad del agua en tiempo real.

Los parámetros que sensa para verificar la calidad del agua son:

- pH
- Óxido disuelto (OD)
- Potencial de oxidación-reducción (ORP)

- Conductividad (Salinidad)
- Turbidez
- Temperatura
- Iones disueltos

La plataforma Smart Water es un nodo de ultra baja potencia, diseñado para usarse en entornos resistentes e implementarse dentro de ciudades inteligentes en lugares de difícil acceso para detectar en tiempo real cambios y riesgos potenciales para la salud pública [25].

2.4.5 Ciudades inteligentes sostenibles.

Según la Unión Internacional de Telecomunicaciones (UIT), en el artículo "Una visión general de las ciudades inteligentes sostenibles y el papel de las tecnologías de la información y comunicación", define que las ciudades inteligentes son aquellas que utilizan tecnologías de la información y comunicación (TIC) para la innovación, con el objetivo de mejorar la calidad de vida, eficiencia de operación y servicios urbanos, garantizando satisfacciones tanto para generaciones presentes y futuras en el aspecto económico, ambiental y sociales [26].

En términos generales, SSC (Sus siglas en inglés Smart Sustainable Cities), se entiende como las iniciativas tecnológicas usadas con el fin de causar un impacto positivo en la vida cotidiana de las personas, contribuyendo a factores ambientales, económicos, sociales, culturales, entre otros propios de una metrópolis. El objetivo que tienen las SSC, es mejorar la vida de los ciudadanos, prestando diferentes servicios como:

- Prestación y acceso a los recursos hídricos y energéticos.
- Transporte y movilidad.
- Educación.
- Medio ambiente.
- Vivienda.
- Subsistencia.

Las TIC, conforman toda clase de plataforma tecnológica, permitiendo superar retos y aprovechar todas las oportunidades que se presenten, impulsando el desarrollo en ciudades y posibilitando la iniciativa de convertirse en ciudades inteligentes y sostenibles [16].

2.4.6 Sistema Inteligente de Transporte

Las ITS[13] , son el conjunto de las tecnologías de nueva generación, aplicadas para el desarrollo de sistemas de transportes, ya sea en aspectos como movilidad, confort, seguridad, entre otras particularidades que presentan los ITS en el transporte habitual. Estos sistemas se pueden ramificar dependiendo del perfil que cumpla el ciudadano, es decir, ciclista, peatón, conductor, pasajero, entre otros, que cambia según el sistema de transporte utilizado. Por otro lado, los sistemas inteligentes de transporte tienen la opción de formar parte en los sistemas de transporte terrestre como movilidad peatonal, movilidad en bicicleta, vehículos particulares y de transporte público [3].
Por ende, todo esto es posible gracias al uso de las herramientas que ofrecen las TIC y por su papel que desempeñan en los ITS, obteniendo beneficios como:

- Ayudar a la disminución de la contaminación ambiental.
- Ahorro de tiempo.
- Ahorro económico a nivel general.
- Seguridad pública.

La principal cualidad respecto al funcionamiento que tienen las ITS son el monitoreo que toman los diferentes dispositivos y el sensado, teniendo algunas aplicaciones como la gestión de tráfico, centralizando información de muchos dispositivos conectados, donde se planean y se toman decisiones para un objetivo social [27].
Basado en lo anterior, ParkUrBike [29] es un sistema de cicloparqueadero que se rige bajo los principios de SSC IoT, él cuenta los servicios de disponibilidad en los espacios de parqueo, identificación de usuarios por espacios de parqueo, de la ubicación del cicloparqueadero y monitoreo ambiental [28].

[13] Del inglés Intelligent Transport System.

2.5 PARKURBIKE

ParkUrBike es un sistema de cicloparqueadero inteligente [29], dividido en tres sistemas, un sistema de disponibilidad, un sistema de control de acceso y un sistema de monitoreo ambiental. El sistema de disponibilidad consta a su vez de un sistema de control de presencia, un sistema de comunicación entre Gateway Intel Edison - repositorio Bucket y un aplicativo en página web [28] (Figura 12).

Figura 12. Sistema de disponibilidad

Fuente: CEA-IoT nodo USTA.

El sistema de control de acceso consta de tres componentes: un diseño de una Interfaz web donde el usuario puede registrarse para tener acceso al sistema; una arquitectura en AWS compuesta por un Api Gateway, IoT Core, Lambda y DynamoDB; y una infraestructura encargada de capturar datos de usuario a través de tarjetas de desarrollo Arduinos, circuitos integrados y solenoides, que serán los encargados de comunicarse entre sí y enviar los datos de los usuarios que marcan el ingreso o retiro de su bicicleta.

Para llevar a cabo este proyecto, se tuvieron en cuenta una serie de requerimientos basados en la necesidad de un sistema de control de acceso que permita identificar, autentificar y dar permisos o autorización a un usuario o grupo de usuarios específicos, que

quieran acceder a un bien específico, mientras se cumplen unos niveles de nivel de seguridad y acceso definidos.

Este sistema de control de acceso consiste en la identificación de un usuario del cicloparqueadero, con el fin de, asociarlo a un espacio de parqueo. Asì, el sistema debe comunicar por algún medio de comunicación efectivo, el usuario asociado a cada espacio de parqueo al gateway del cicloparqueadero. Y posteriormente, la información enviada por el sistema de control de acceso será publicada en una interfaz web.

En el presente documento se realiza una descripción detallada de todo el sistema de control de acceso, incluidos cada uno de sus componentes.

3 REQUERIMIENTOS DEL SISTEMA DE CONTROL DE ACCESO

Los casos de uso (Anexo A) contemplados en este sistema son: identificar al usuario, autenticar al usuario y otorgar permisos de acceso al usuario ya sea para el ingreso o retiro de la bicicleta.

Para cumplir con estos requerimientos, el sistema de control de acceso debe cumplir con las siguientes especificaciones:

- RQ1: El sistema debe de tener un control de entrada y salida del usuario.
- RQ2: El sistema debe permitir el registro de usuarios mediante una aplicación web.
- RQ3: El sistema debe tener un método de registro, dado el caso si el usuario no tiene acceso a la interfaz web.
- RQ4: El sistema debe tener métodos para la gestión de asignar o liberar espacios de parqueo.
- RQ5: El sistema debe identificar, autentificar y autorizar al usuario el ingreso al sistema.
- RQ6: El sistema debe tener un método de actualización de información del usuario.
- RQ7: El sistema debe guardar información del usuario del cicloparqueadero, tales como:
    - Número de identificación
    - Nombre
    - Apellido
    - Código institucional
    - Correo institucional
    - Facultad
    - Tipo de usuario
    - Marca de la bicicleta
    - Color de la bicicleta
- RQ8: El sistema debe guardar la información de uso del ciclo parqueadero, como:
    - Hora de entrada de bicicletas.
    - Hora de salida de bicicletas.

- Estado de los espacios de parqueo.

➢ RQ9: El sistema debe brindar seguridad a las bicicletas.
➢ RQ10: El sistema debe generar reportes estadísticos.

Con el fin de dar cumplimiento a estos requerimientos, la arquitectura planteada se presenta en la Figura 13. Los detalles de esta arquitectura se desarrollan en los capítulos 3 y 4.

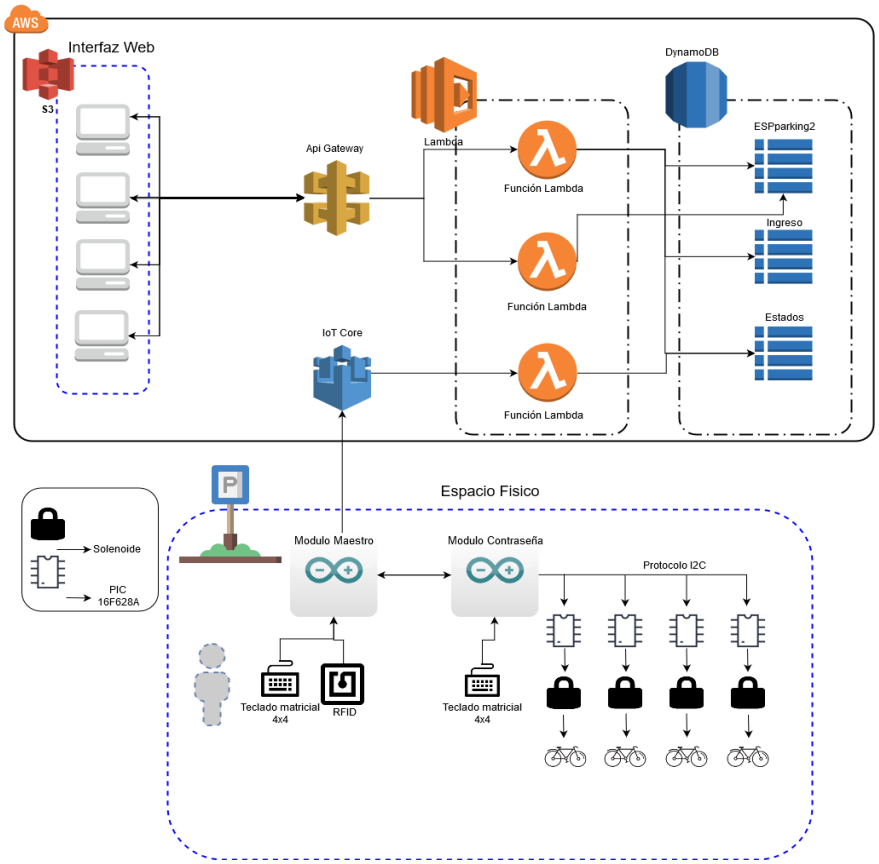


Figura 13. Sistema de control de acceso.

Fuente: Autores.

Este proyecto debe ser implementado bajo el marco metodológico elegido (Capítulo 1.3). Cada sprint debe realizarse semanalmente donde el grupo de trabajo debe revisar cada tarea del backlog[14] que se encuentra marcada “en proceso” y se verifica que estas tareas pueden ser marcadas como “Finalizadas” y para las que se encuentran marcadas como “Cosas por hacer” se asigna un encargado que debe cumplir con la esta tarea con un tiempo de realización inferior o igual al tiempo en que se realizaría el siguiente sprint.

Se realizó una tabla (Anexo B) donde se muestra el backlog implementado para el desarrollo del proyecto.

[14] Lista de tareas necesarias para el desarrollo de un producto

4 INFRAESTRUCTURA DEL SISTEMA DE CONTROL DE ACCESO

Los detalles presentados en este capítulo responden a los siguientes requerimientos:

- RQ1: El sistema debe de tener un control de entrada y salida del usuario.
- RQ3: El sistema debe tener un método de registro, dado el caso si el usuario no tiene acceso a la interfaz web.
- RQ4: El sistema debe tener métodos para la gestión de asignar o liberar espacios de parqueo.
- RQ5: El sistema debe identificar, autentificar y autorizar al usuario el ingreso al sistema.
- RQ9: El sistema debe brindar seguridad a las bicicletas.

4.1 DISEÑO

La infraestructura del sistema de control de acceso se diseñó con base a cumplir a los requerimientos seleccionados anteriormente y consta de tres módulos: el Módulo Maestro se encargará de la asignación de los espacios de parqueo, registro del usuario en caso que no tenga acceso a la interfaz web y la lectura de las tarjetas RFID que serán los carnets de la universidad. Módulo Contraseña tiene como función principal validar la contraseña para liberar los espacios de parqueo y Módulos Esclavo el cual estarán conectados a un bus de datos unidireccional, se encargarán de la seguridad de las bicicletas en los espacios de parqueo donde cada uno cumplirá una función específica en el sistema ubicados en el sector de Espacio Físico (ver figura 14). Para la comunicación entre los módulos mencionados anteriormente se determinó que va a ser por medio de cable UTP con interfaces físicas RJ45, la razón de usar cable UTP con interfaces físicas RJ45 en el sistema es debido por su practicidad, también como es de uso libre para estructura de cableado, el orden no importa por ende en el desarrollo se define la función de cada cable, porque es el más popular del mercado y es el más fácil de encontrar repuestos para el mantenimiento y la más importante es que hoy en día las redes actuales utilizan ethernet.

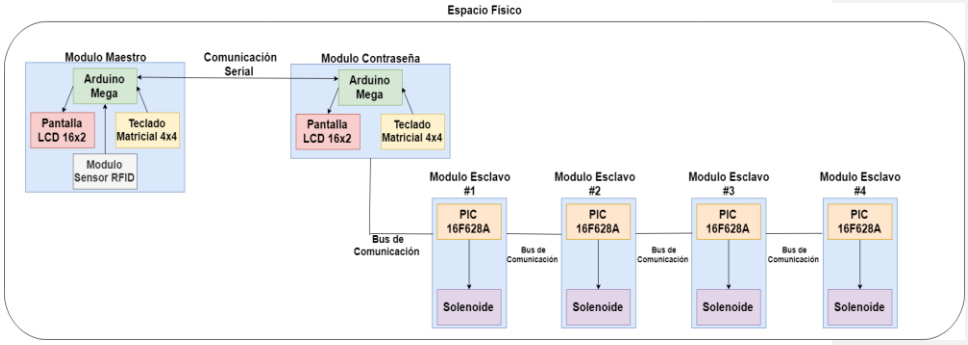


Figura 14. Sector Espacio Físico del sistema de control de acceso

Fuente: Autores.

4.1.1 Módulo Maestro

El Módulo Maestro es el encargado de determinar si el usuario va a ingresar la bicicleta al sistema otorgando un espacio de parqueo o a retirar la bicicleta del sistema asignando una contraseña aleatoria comunicando al Módulo Contraseña, de igual forma el Módulo Maestro tendrá acceso a AWS donde podrá inscribir al usuario en el sistema, reconocer al usuario, disponibilidad de los espacios de parqueo y la hora en la que el usuario ingresa al sistema.

Para lograr lo anterior, el módulo maestro se diseñó a partir de una matriz en la cual se almacenará la información del uso de los espacios de parqueo del cicloparqueadero. Además el módulo maestro trabajara dependiendo de dos estados: el primer estado consiste en determinar cuál es el motivo de que el usuario ingrese al sistema, es decir si viene a ingresar la bicicleta o a retirarla, mientras el segundo estado consiste en un breve registro en el sistema, para ello, se le pedirá por medio de la pantalla LCD que digite en el teclado matricial un número entre 1 y 2, el valor número 1 es para ingresar o retirar la bicicleta al sistema y el número dos para registrarse en el sistema dado el caso de que el usuario ingrese al espacio físico y no esté registrado. Esto sucederá al momento que el usuario acerque el carnet de la universidad a un módulo sensor RFID.

4.1.2 Módulo Esclavo

Para el **Módulo Esclavo** se diseñó un prototipo de candados inteligentes conectados a un bus comunicación, con el objetivo de brindar seguridad a los usuarios del cicloparqueadero,

donde cada uno de ellos se accionará después de que el sistema identifique al usuario que va a ingresar al Espacio Físico, lo autentifique y le otorgue permisos al usuario para el ingreso al espacio físico, ya sea para guardar o retirar la bicicleta.

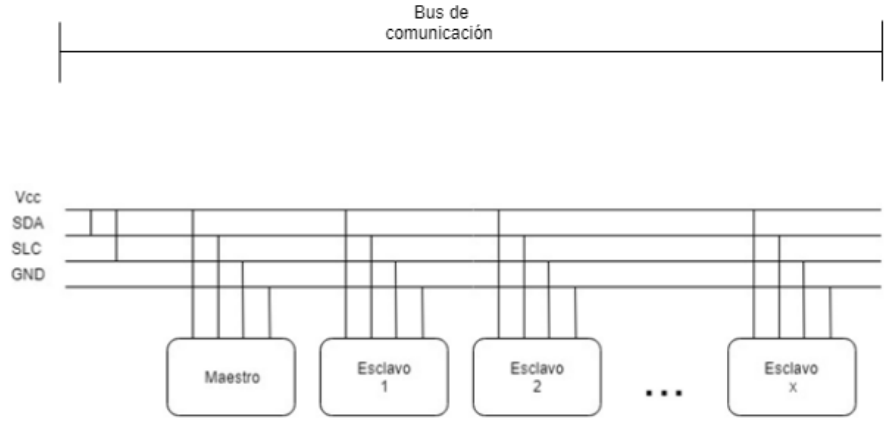


Figura 15. Diagrama del bus de comunicación del sistema control de acceso

Fuente: Autores.

El bus de comunicación (Figura 15) juega un papel fundamental, ya que es el encargado de la alimentación de los Módulos Esclavos, de igual forma cuenta con el protocolo I2C[15] que es un protocolo de comunicación serial entre dispositivos permitiendo tener múltiples esclavos y también está compuesta con las conecciones a tierra [30].

4.1.3 Módulo Contraseña

El Módulo Contraseña tiene como función principal validar la contraseña que ingrese el usuario con la que recibe por puerto serial del Módulo Maestro. El Módulo Contraseña se diseñó con base a los requerimientos mencionados anteriormente y actuará dependiendo del estado el cual será determinado por el Módulo Maestro, las cuales son:

- La apertura del Módulo Esclavo asignado por el Módulo Maestro, habilitando el espacio de parqueo permitiendo al usuario colocar la bicicleta.

[15] Por sus siglas en inglés Inter-Integrated Circuit

- El proceso de validación de la contraseña ingresada por el usuario con la contraseña asignada por el Módulo Maestro, con el fin de habilitar el espacio de parqueo permitiendo al usuario retirar la bicicleta.

A continuación, se realizará una descripción detallada del desarrollo de cada componente del sector Espacio Físico del sistema de control de acceso donde se comenzará con el diseño del Módulo Maestro, seguido de los Módulos Esclavos y finalizando el Módulo Contraseña.

4.2 DESARROLLO

4.2.1 Módulo Maestro

El desarrollo del Módulo Maestro está basado en el diseño anteriormente presentado y en el cumpliendo de los requerimientos planteados en el capítulo 2. Para ello, el Módulo Maestro maneja dos estados con el fin de determinar si el usuario va a ingresar o retirar la bicicleta del espacio físico, digitando un número entre 1 y 2. En el Módulo Maestro se encuentra una matriz donde en ella se almacenará la información respecto uso de los espacios de parqueo del cicloparqueadero y el dato de la tarjeta de identificación, vale especificar que dichas tarjetas poseen una memoria, donde el bloque 0 contiene el ID o dato único de identificación de usuario, el cual se capturará y almacenará en una base de datos en AWS.

En el caso de que el usuario digite el número 1, el Módulo Maestro hará lo siguiente:

- Revisar la matriz si el dato de la tarjeta de identificación RFID que leyó se encuentra guardado en ella
- Si el dato de la tarjeta de identificación RFID se encuentra guardado en la matriz es porque el usuario va a retirar la bicicleta.
- Si el dato de la tarjeta de identificación RFID no se encuentra guardado en la matriz es porque el usuario va a retirar la bicicleta.

La función que tiene la matriz además de guardar los datos de las tarjetas de identificación RFID, es gestionar la asignación de los espacios de parqueo.

Figura 16. Dispositivo Módulo Maestro

Fuente: Autores.

Si el usuario va a ingresar al sistema, se mostrará por una pantalla LCD el espacio de parqueo asignado, se guardará el dato de la tarjeta de identificación RFID en la matriz con el espacio de parqueo asignado y por un puerto serial que conectara al Módulo Contraseña, se enviará el valor para el valor en hexadecimal del Módulo Esclavo el cual le fue asignado al usuario con el fin de que el espacio de parqueo abra permitiendo al usuario estacionar la bicicleta, al mismo tiempo a través del protocolo MQTT este módulo envía un JSON al AWS IoT Core que ejecuta una función Lambda para su registro en una tabla de DynamoDB donde se guardan los registros de las personas que ingresan o retiran su bicicleta y las personas que se registran en el sistema. El módulo procederá a crear una contraseña de números aleatorios que se mostrará en pantalla y, para terminar, un arreglo con unos datos del JSON se enviará por el puerto serial que comunica al Módulo Contraseña.

Por el contrario, si el usuario va retirar la bicicleta del espacio de parqueo, teniendo en cuenta que el dato de identificación RFID ya se encuentran guardados en la matriz, el módulo tomará el dato de la tarjeta de identificación RFID y hará una revisión en la matriz

para corroborar que el dato de identificación RFID se encuentre guardado en ella. Después de verificar que el usuario va a retirar la bicicleta, llenará un arreglo con el dato de la tarjeta de identificación RFID y el espacio de parqueo asignado, este arreglo a través del protocolo MQTT enviará un JSON que pasara por Shadow de IoT que ejecuta una función Lambda para su registro en una tabla en DynamoDB, a su vez otro arreglo con los datos del espacio de parqueo asignado y la contraseña aleatoria, se enviará por el puerto serial que comunica al Módulo Contraseña, a la espera de que el usuario ingrese la contraseña correctamente para poder limpiar la matriz que se encuentra guardado en dicho módulo.

Para el caso en que se digite el número 2, es para el momento donde el usuario ingrese al sistema y no se encuentre registrado, donde el módulo tomará el dato de la tarjeta de identificación RFID, a través del protocolo MQTT enviará un JSON que pasara por Shadow de IoT que ejecuta una función Lambda para su registro en una tabla en DynamoDB.

4.2.2 Gateway Módulo Maestro

Siguiendo la recomendación de Gartner para el diseño e implementación de una solución IoT, el IoT Edge Platform de la infraestructura se encuentra en el mismo nodo donde el controlador del sistema está alojado, este dispositivo es un ESP 8266 Node MCU con un framework que puede ser trabajado con el lenguaje arduino, esta tarjeta cumple la función de ser el gateway entre la arquitectura en la nube y los dispositivos Módulos Esclavos.

El protocolo de comunicación que este dispositivo para la conexión con la arquitectura en la nube es MQTT donde, según el protocolo, deben existir unos *topic* de sesión para realizar la conexión directa y exitosa con el servidor, como también el servicio en la nube que se utilizó para realizar la conexión directa genera unos ***topic's*** adicionales para autenticar el dispositivo que se está conectando, el código que se ingresó a la tarjeta fue totalmente lenguaje Arduino importando el SDK que AWS provee para el desarrollo de soluciones IoT, incluyendo un archivo de autenticación que contiene los ***topic's*** necesarios para conectarse al servidor en la nube.

Este dispositivo haciendo edge computing informa el estado actual del sistema; cuando un usuario ingresa, retira o se registra en el sistema, este dispositivo recibe los parámetros generados y genera un topic que se envía al IoT Core para que se procesen los datos dependiendo del tipo de informe generado.

4.2.3 Módulo Esclavo

Los Módulos Esclavos son aquel componente del sistema de control de acceso que serán ubicados en los estacionamientos en el cual el usuario colocara la bicicleta y su función principal es comparar condiciones provistas por el Módulo Contraseña para activar o desactivar la cerradura de un estacionamiento. Los componentes del circuito de los candados inteligentes son:

- Puertos RJ-45 hembra.
- Circuito puente H.
- PIC 16F628A.
- MAX 465.

Figura 17. Dispositivo Módulo Esclavo

Fuente: Autores.

Para el desarrollo, se utilizó el montaje de un puente H con el objetivo de accionar el solenoide, es decir, como el solenoide trabaja con una alimentación de 12V y la salida del microcontrolador pic 16F628A es un pulso de 5V, por ende el pulso del microcontrolador no tiene el voltaje suficiente para accionar el solenoide, para ello, el motivo de uso del punte H es que por medio del pulso que tiene la salida del microcontrolador permite el paso de una segunda fuente de alimentación de 12V, como se puede observar en la siguiente imagen (Figura 18).

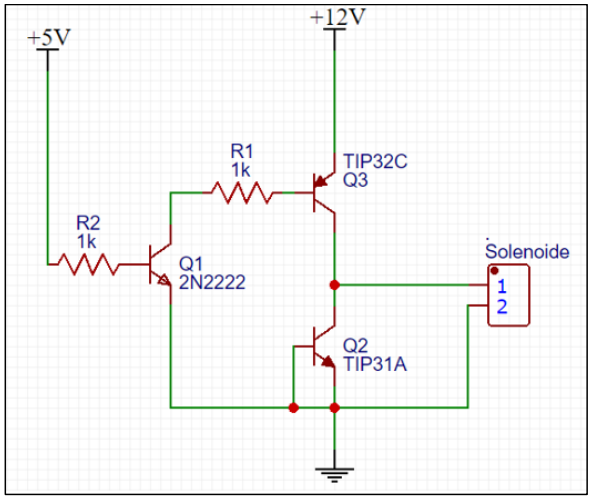


Figura 18. Montaje de medio puente H para los Módulos Esclavos

Fuente: Autores.

El integrado PIC 16F628A (Figura 19) es un microcontrolador y el componente más importantes del Módulo Esclavo, ya que está compuesto por una CPU y unidades de memoria RAM y ROM, teniendo como funcionalidad poder almacenar datos o programas en él, el microcontrolador fue programado en lenguaje ensamblador o assembler.

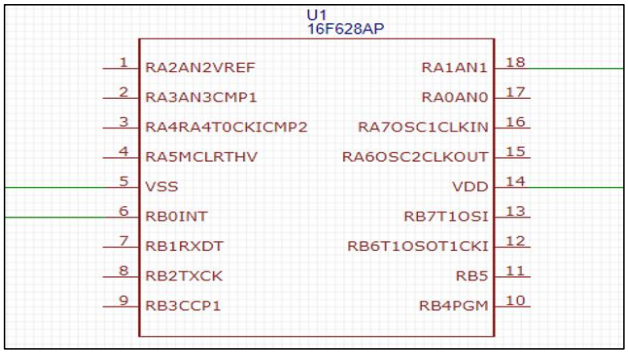


Figura 19. Microcontrolador PIC 16F628A

Fuente: Autores.

Se utilizó este microcontrolador por su tamaño debido a que es pequeño, por su valor económico ya que en el mercado existe gran variedad de microcontroladores y diversos precios, además posee una arquitectura RISC[16] ya que es una instrucción sencilla la que se ejecuta y por recomendación de docentes que tienen experiencia con microcontroladores.

Para la comunicación entre los Módulos Esclavos y el Módulo Contraseña, se escogió el protocolo I2C, es un protocolo de comunicación serial que consiste en conectar varios dispositivos a un maestro todos al mismo tiempo siendo el maestro el que distribuye la información y los esclavos trabajando con la misma señal de reloj reciben la información que se distribuyó, se utilizó este protocolo ya que son múltiples Módulos Esclavos los que van a estar conectados al Módulo Contraseña, quien es el encargado de enviar el dato para que el microcontrolador ejecute la condición con la cual está programado [30].

El MAX485 es un integrado comúnmente utilizado para el protocolo I2C por su amplia capacidad de poder tener varios dispositivos esclavos conectados a un maestro, por ende, se utilizó para el desarrollo del Módulo Esclavo.

Por último, otro componente que utilizamos para el desarrollo del Módulo Esclavo fue la interfaz física RJ-45, es un protocolo de interfaz física usado comúnmente en redes de cableado estructurado, para este caso, se aplicó la configuración de cableado “568-B” cableado directo, donde cada cable del par trenzado del cable UTP cumple una función que son las siguientes:

Tabla 1. Tabla de relación Color del cable con Función de un par trenzado para el sistema de control de acceso

Color del cable	**Función**
Blanco/Naranja	Tierra (-)
Naranja	5V (+)
Blanco/Verde	
Verde	

[16] Por sus siglas en inglés Reduced Instruction Set Computer

Blanco/Azul	Puerto 7 del MAX 485
Azul	Puerto 6 del MAX 485
Blanco/Café	12V(+)
Café	

Fuente: Autores.

El montaje del circuito se puede apreciar en la siguiente imagen (Figura 20).

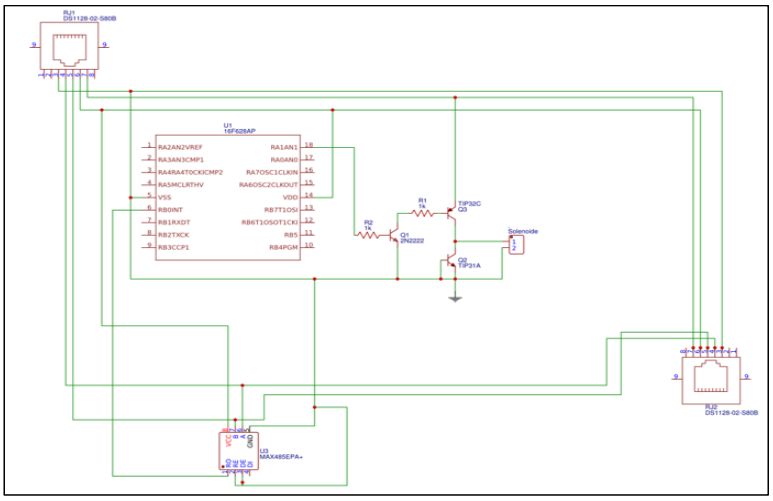


Figura 20. Montaje del circuito de los Módulos Esclavos

Fuente: Autores.

Y para finalizar, con la ayuda de un software online con nombre Easyeda, se elaboró PCV de dos capas para todo el circuito completo del Módulo Esclavo (Figura 21).

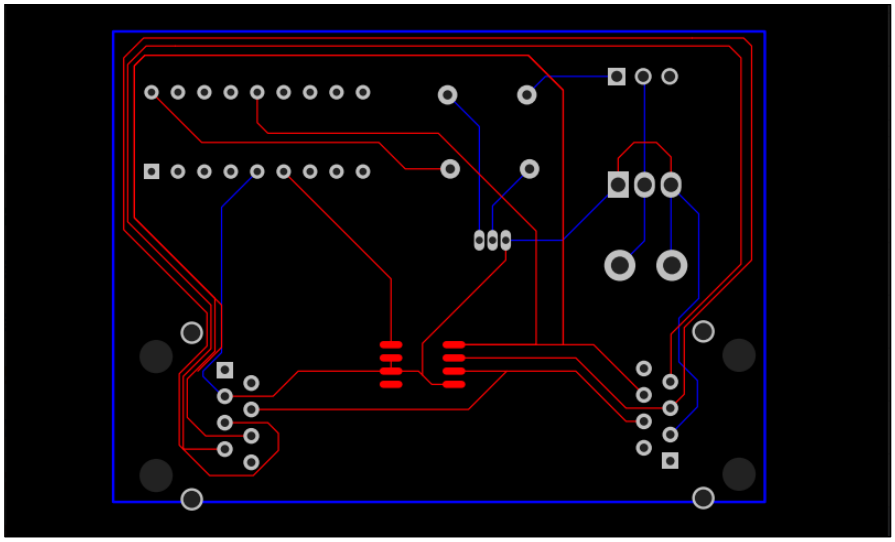


Figura 21. PCB de los Módulos Esclavos

Fuente: Autores

4.2.4 Módulo Contraseña

El desarrollo del Módulo Contraseña fue con base a las dos funciones planteadas en su diseño y bajo los requerimientos mencionados al principio del capítulo, las acciones que ejecutará el Módulo Contraseña dependiendo del estado el cual será determinado por el Módulo Maestro son:

- Ingreso al sistema. Cuando el usuario va a ingresar al sistema, recibe por puerto serial el espacio asignado por el módulo maestro y por el bus I2C envía el valor con el que se programó el Microcontrolador 16F628A para la apertura del espacio de parqueo para que el usuario pueda ingresar y poder colocar la bicicleta en el espacio de parqueadero asignado por el Módulo Maestro.
- Salida del sistema (Retiro de bicicleta). Si el usuario va a retirar la bicicleta del sistema, El Módulo Contraseña recibe por puerto serial la contraseña proveniente del Módulo Maestro y la compara con el valor digitado en el teclado matricial que es la clave asignada al espacio de parqueo, donde va a tener un máximo de 3 intentos para ingresar la clave en caso de que el usuario se equivoque al digitarla, en el

momento de que el usuario ingrese la clave de manera correcta, el Módulo Contraseña enviará el valor respectivo del candado candado inteligente para activarla y poder dejar al usuario retirar la bicicleta sin ningún problema, de lo contrario si no se cumple la condición no hará nada.

Figura 22. Dispositivo Módulo Contraseña

Fuente Propia.

El Módulo Contraseña está compuesto de 5 elementos que son los siguientes:

- Módulo adaptador I2C para pantallas LCD
- Teclado matricial 4X4
- Módulo RFID
- Pantalla LCD 16x2
- Arduino Mega

El Módulo Contraseña recibirá por puerto serial un arreglo con información proveniente del Módulo Maestro, dicho arreglo tendrá la información del espacio de parqueo asignado y la contraseña asignada, una vez recibida dicha información, se guardará en una matriz, para cuando el usuario digite la contraseña en el Módulo Contraseña, se realice la comparación con la contraseña recibida del Módulo Maestro.

4.3 VALIDACIÓN

Para la validación de la infraestructura del sistema de control de acceso se verificó el funcionamiento haciendo una serie de pruebas organizadas en una tabla donde:

- Se enumera la prueba.
- Ingresa el nombre del sector al que se va a realizar.
- Se escribe una descripción muy breve de lo que se va a realizar.
- El resultado esperado.
- El resultado obtenido.
- La prueba fue exitosa o fallida.

4.3.1 Validación Módulo Maestro

Las pruebas realizadas para la validación del funcionamiento del Módulo Maestro se pueden evidenciar en la siguiente tabla.

Tabla 2. Módulo Maestro

PRUEBAS PARA LA VALIDACIÓN DEL SISTEMA DE CONTROL DE ACCESO PARKURBIKE					
#	**COMPONENTE**	**DESCRIPCIÓN**	**RESULTADO ESPERADO**	**RESULTADO OBTENIDO Y OBSERVACIONES**	**TIPO DE RESULTADO**
1	Módulo Maestro	Comunicación con el Módulo Contraseña.	Mostrar el arreglo que recibe por puerto serial en monitor serie de Arduino.	En el monitor serie se mostró el arreglo proveniente del Módulo Maestro	Exitoso.
2	Módulo Maestro	Conexión del teclado matricial al Arduino.	Mostrar en monitor serie de Arduino lo que se ingrese en el teclado.	En el monitor serie se mostró lo digitado en el teclado.	Exitoso.
3	Módulo Maestro	Conexión de la pantalla LCD a Arduino.	Mostrar los datos que se ingresen en el monitor serie.	No se pudo conectar la LCD por falta de puertos digitales.	Fallido.
4	Módulo Maestro	Conexión de la pantalla LCD al	Mostrar los datos que se ingresen en	Se mostró en la pantalla LCD lo	Exitoso.

		Arduino por medio de I2C.	el monitor serie.	datos que se ingresaron en el monitor serie.	
5	Módulo Maestro	Conexión del sensor RFID al Arduino.	Mostrar en monitor serie el número de identificacion RFID de la tarjeta.	Se mostró en monitor serie el número de identificación RFID de la tarjeta.	Exitoso
6	Módulo Maestro	Asignación de espacios de parqueo en la matriz del Módulo Maestro y creación de contraseña aleatoria.	Mostrar en monitor serie el espacio de parqueo asignado y la contraseña creada.	Se asignaron correctamente los espacios de parqueo.	Exitoso.
7	Módulo Maestro	Envío del arreglo con la información del espacio de parqueo y la contraseña aleatoria	Mostrar en monitor serie del Módulo contraseña el arreglo con la información del espacio de parqueo y la contraseña	Se mostró en el monitor serie del Módulo contraseña el arreglo con la información del espacio de parqueo y la contraseña	Exitoso.
8	Módulo Maestro	Envío del arreglo con la información del espacio de parqueo y la contraseña aleatoria	Mostrar en una tabla DynamoDB el registro de ingreso	Se mostró en la tabla DynamoDB el registro de ingreso.	Exitoso.
9	Módulo Maestro	Inscripción del usuario al sistema.	Inscribir al usuario que ingresa al sistema y no se encuentre registrado, en una tabla de DynamoDB.	se inscribió el usuario que ingresa al sistema y no se encuentre registrado en DynamoDB.	Exitoso.

Fuente: Autores

El problema que se presentó en el momento de realizar pruebas para su respectiva validación, fue la falta de puertos digitales en la placa de arduino, ya que la conexión de la

pantalla LCD del Módulo Maestro requiere de muchas conexiones en dichos puertos y la disponibilidad es poca. La solución que se llevó a cabo fue utilizar el Módulo adaptador de LCD a I2C, con el fin de utilizar el utilizar la pantalla LCD con solo dos puertos digitales.

4.3.2 Validación Módulo Contraseña

Para la validación el Módulo Contraseña, se utilizaron los 4 módulos esclavos, el cual estaban conectados en serie por medio de un bus de comunicación, donde el Módulo Contraseña transmite información por medio del bus de comunicación con destino a los Módulos Esclavos, quienes están escuchando a la espera de ser accionados.

Las pruebas realizadas para la validación del funcionamiento del Módulo Contraseña se pueden evidenciar en la siguiente tabla.

Tabla 3. Módulo Contraseña

PRUEBAS PARA LA VALIDACIÓN DEL SISTEMA DE CONTROL DE ACCESO PARKURBIKE					
#	**COMPONENTE**	**DESCRIPCIÓN**	**RESULTADO ESPERADO**	**RESULTADO OBTENIDO Y OBSERVACIONES**	**TIPO DE RESULTADO**
1	Módulo Contraseña	Comunicación con el Módulo Maestro.	Mostrar el arreglo que recibe por puerto serial en monitor serie de Arduino.	En el monitor serie se mostró el arreglo proveniente del Módulo Maestro	Exitoso.
2	Módulo Contraseña	Conexión del teclado matricial al Arduino.	Mostrar en monitor serie de Arduino lo que se ingrese en el teclado.	En el monitor serie se mostró lo digitado en el teclado.	Exitoso.
3	Módulo Contraseña	Conexión de la pantalla LCD a Arduino.	Mostrar los datos que se ingresen en el monitor serie.	No se pudo conectar la LCD por falta de puertos digitales en la placa de Arduino.	Fallido.
4	Módulo Contraseña	Conexión de la pantalla LCD al Arduino por medio	Mostrar los datos que se ingresen en el monitor serie.	Se mostró en la pantalla LCD lo datos que se ingresaron en el	Exitoso.

		de I2C.		monitor serie.	
5	Módulo Contraseña	Conexión de la pantalla LCD con el teclado matricial.	Mostrar en pantalla LCD los datos ingresados en el teclado matricial.	Se mostró en pantalla LCD los datos ingresados en el teclado matricial.	Exitoso.
6	Módulo Contraseña	Proceso de asignación de usuarios en la matriz.	Asignar el arreglo proveniente del Módulo Maestro en una matriz.	Se asignaron correctamente los arreglos provenientes del Módulo Maestro.	Exitoso.
7	Módulo Contraseña	Validación de la contraseña enviada en el arreglo y asignada en la matriz con los datos ingresados en el teclado matricial, teniendo un máximo de 3 intentos.	Mostrar en un osciloscopio la salida del Módulo Contraseña cuando el usuario ingrese la contraseña correctamente.	Se mostró en el osciloscopio el valor en hexadecimal en la salida del Módulo Maestro.	Exitoso.

Fuente: Autores

De igual forma, el Módulo Contraseña presentó el mismo problema que se presentó con el Módulo Maestro en el momento de realizar pruebas para su respectiva validación, fue la falta de puertos digitales en la placa de arduino, ya que la conexión de la pantalla LCD del Módulo Maestro requiere de muchas conexiones en dichos puertos y la disponibilidad es poca. La solución utilizada en el Módulo Maestro se aplicó de igual forma en el Módulo Contraseña, basada en utilizar el Módulo adaptador de LCD a I2C, con el fin de utilizar el utilizar la pantalla LCD con solo dos puertos digitales.

4.3.3 Validación Módulo Esclavo

Las pruebas realizadas para la validación del funcionamiento del Módulo Esclavo se pueden evidenciar en la siguiente tabla.

Tabla 4. Módulos Esclavos

PRUEBAS PARA LA VALIDACIÓN DEL SISTEMA DE CONTROL DE ACCESO PARKURBIKE

#	COMPONENTE	DESCRIPCIÓN	RESULTADO ESPERADO	RESULTADO OBTENIDO Y OBSERVACIONES	TIPO DE RESULTADO
1	Módulos Esclavos	Programa el microcontrolador en lenguaje C que encienda un LED.	Encender y apagar un LED cada segundo.	Se encendió y apago el LED correctamente.	Exitoso.
2	Módulos Esclavos	Programar en lenguaje C condición en el microcontrolador.	Encender y apagar un LED si el valor hexadecimal que entra al microcontrolador es igual al valor hexadecimal programado en él.	No se encendió el LED.	Fallido.
3	Módulos Esclavos	Programar en lenguaje Assembler condición en el microcontrolador.	Encender y apagar un LED si el valor hexadecimal que entra al microcontrolador es igual al valor hexadecimal programado en él.	El LED se encendió y apagó.	Exitoso.
4	Módulos Esclavos	Montaje de medio Puente H con el solenoide.	Ingresando un pulso de 5V a un circuito compuesto por la mitad de un puente H para activar el solenoide.	Se activó el solenoide ingresando 5V en el circuito	Exitoso.
5	Módulos Esclavos	Unión del microcontrolador con el circuito	Activar el solenoide una vez se cumpla la condición del microcontrolador	Se activó el solenoide al momento de que se cumplio la condicion.	Exitoso.
6	Módulos Esclavos	Interacción del Módulo Contraseña con un Módulos Esclavos.	Activar el solenoide por medio del valor en hexadecimal proviniendo del	Se activó el solenoide.	Exitoso.

			Módulo Contraseña.		
7	Módulos Esclavos	Interacción del Módulo Contraseña con más de un Módulo Esclavo.	Activar varios solenoides por medio del valor en hexadecimal proviniendo del Módulo Contraseña.	No se activó el solenoide.	Fallido.
8	Módulos Esclavos	Aplicación de Protocolo I2C para la interacción del Módulo Contraseña con más de un Módulo Esclavo.	Activar varios solenoides por medio del valor en hexadecimal proviniendo del Módulo Contraseña, aplicando un bus de datos unidireccional.	Se activó el solenoide.	Exitoso
9	Módulos Esclavos	Aplicación de protocolo de conexión tipo Ethernet entre los Módulos Esclavos y Módulo Contraseña.	Activar varios solenoides por medio del valor en hexadecimal proviniendo del Módulo Contraseña, aplicando un bus de datos unidireccional.	Se activó el solenoide.	Exitoso

Fuente: Autores

El problema que se presentó en el momento de realizar pruebas para su respectiva validación, fue la comunicación del Módulo Contraseña con los Módulos Esclavos, una vez estando conectados, cuando el Módulo Contraseña enviaba el valor para abrir un Módulo Esclavo en específico no funcionaba, la solución a dicho problema fue aplicar el protocolo de comunicación I2C, permitiendo tener múltiples Módulos esclavos a un maestro que en este caso es el Módulo Contraseña, este protocolo de comunicación se agregó al bus de comunicación junto a las fuentes de alimentación y la conexión a tierra de los Módulos Esclavos.

Dado el caso de que se desconecte un Módulo Esclavo, se perderá la comunicación de los demás Módulos Esclavos, ya que los Módulos Esclavos estarán conectados en serie como se puede apreciar en la Figura 14, este factor problema no se pudo solucionar debido a que el puerto hembra del RJ45 está integrado al case de los Módulos Esclavo.

Dada la circunstancia de que se presente una falla eléctrica en el espacio físico, los Módulos Esclavos se mantendrán cerrados, ya que no habrá una fuente quien los haga accionar, como se mencionó anteriormente, para que el solenoide accione se necesita una fuente que lo alimente, si se llega a presentar una desconexión eléctrica las bicicletas permanecerán seguras.

5 ARQUITECTURA EN LA NUBE DEL SISTEMA DE CONTROL DE ACCESO

Los detalles presentados en este capítulo responden a los siguientes requerimientos:

- RQ2: El sistema debe permitir el registro de usuarios mediante una aplicación web.
- RQ6: El sistema debe tener un método de actualización de información del usuario.
- RQ7: El sistema debe guardar información del usuario del cicloparqueadero, tales como:
  - Número de identificación
  - Nombre
  - Apellido
  - Código institucional
  - Correo institucional
  - Facultad
  - Tipo de usuario
  - Marca de la bicicleta
  - Color de la bicicleta
- RQ8: El sistema debe guardar la información de uso del ciclo parqueadero, como:
  - Hora de entrada de bicicletas.
  - Hora de salida de bicicletas.
  - Estado de los espacios de parqueo.
- RQ10: El sistema debe generar reportes estadísticos.

5.1 DISEÑO

Cumpliendo con los requerimientos y siguiendo el modelo de referencia de Gartner, el diseño del Web Application (WAPP), el IoT Platform Repository (IoT PR) y el IoT Edge Platform (IoT EP) para la arquitectura en la nube del proyecto se diseñó como se muestra en la siguiente figura (Figura 22).

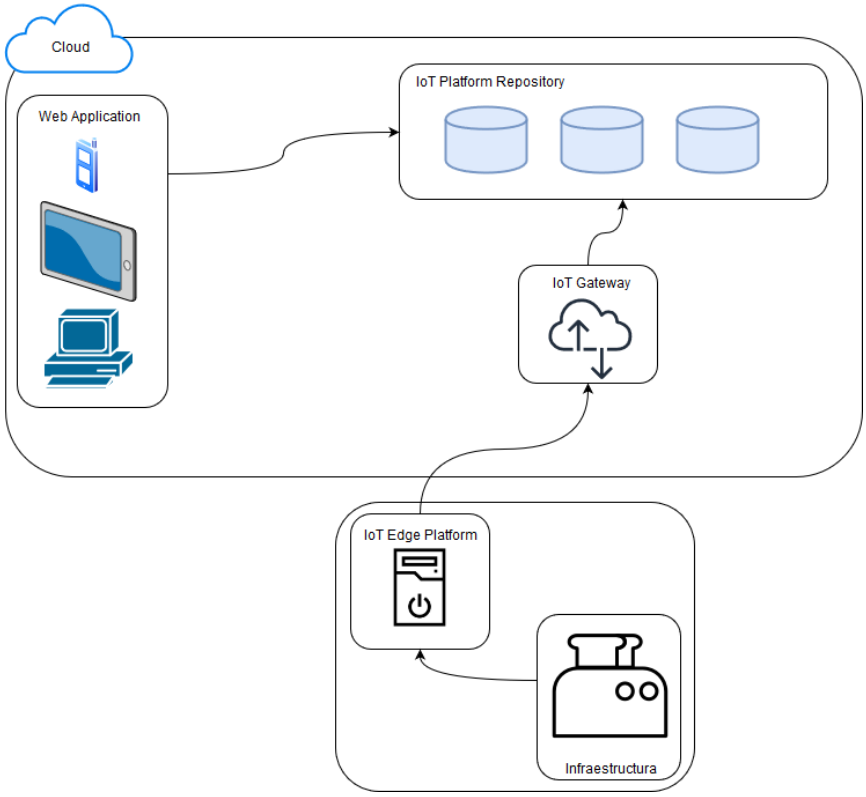


Figura 23. Diseño Arquitectura en la nube

Fuente: Autores

La arquitectura se diseñó en la nube de AWS, debido a que presenta muchos beneficios que la hacen sobresalir con respecto a las demás nubes, como su facilidad de uso, su flexibilidad y su seguridad, añadiendo que AWS es el líder mundial en servicios en la nube, con el 51% del mercado mundial. Además, presta servicios a grandes empresas como Rappi, Bancolombia, Caracol Tv y Davivienda; para finalizar, la Facultad de Ingeniería de Telecomunicaciones de la Universidad Santo Tomás es la primera academia AWS del país.

5.1.1 Web Application

El diseño para el frontend de la WAPP consiste en una interfaz web con 4 ventanas que brindan al usuario: información sobre el servicio que ParkUrBike ofrece; la posibilidad de registrarse en el sistema por medio de un formulario, y una interfaz para reportes en tiempo real sobre el estado actual del servicio.

Para el diseño se tuvo en cuenta la facilidad al momento en que un usuario interactúe con la interfaz web de forma intuitiva, el diseño del frontend de la WAPP consta de las siguientes ventanas:

- Ventana de inicio: Descripción del servicio y responsables del desarrollo.
- Ventana de inscripción: Formulario de registro de un nuevo usuario.
- Ventana de dashboard: Reportes con datos del sistema (*e.g.* uso del sistema, hora de ingreso salida del cicloparqueadero, marcas de las bicicletas registras).
- Ventana de Políticas de privacidad: Marco legal que rige el uso del sistema.

Los diseños (Figura 23, 24, 25, 26) se crearon en Balsamiq, una herramienta en línea para realizar esquemas funcionales para la creación de una aplicación móvil o web, están basados bajo la misma idea de que sean intuitivos para cualquier usuario.

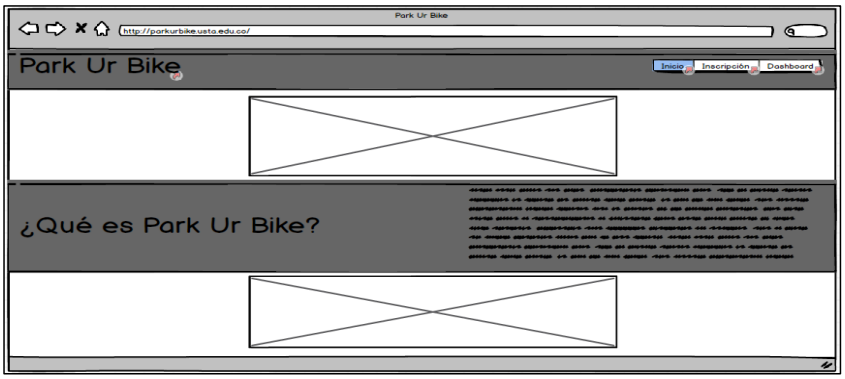


Figura 24. Diseño de la sección Inicio de la Interfaz Web.

Fuente: Autores

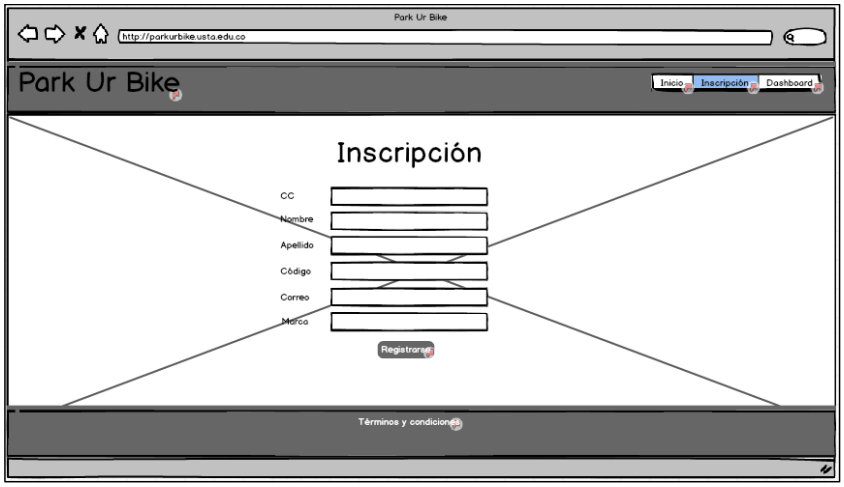


Figura 25. Diseño de la sección Inscripción de la Interfaz Web

Fuente: Autores

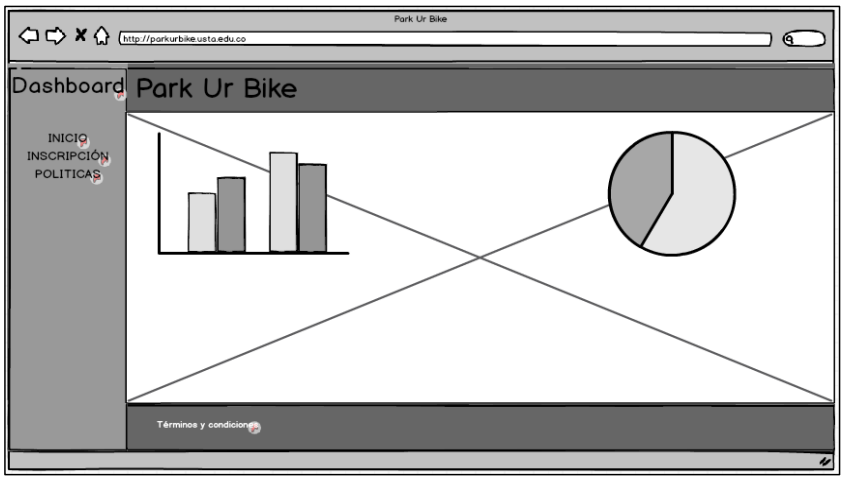


Figura 26. Diseño de la sección DashBoard de la Interfaz Web

Fuente: Autores

Figura 27. Diseño del espacio donde estarán las políticas de privacidad de la Interfaz Web

Fuente: Autores

El procedimiento que la WAPP debe hacer cuando un usuario va a registrar sus datos personales debe verificar que todos los datos ingresados sean digitados correctamente, como también validar que todos los campos se encuentren llenos antes de enviar la solicitud que registre los usuarios en la base de datos de usuarios (Figura 27).

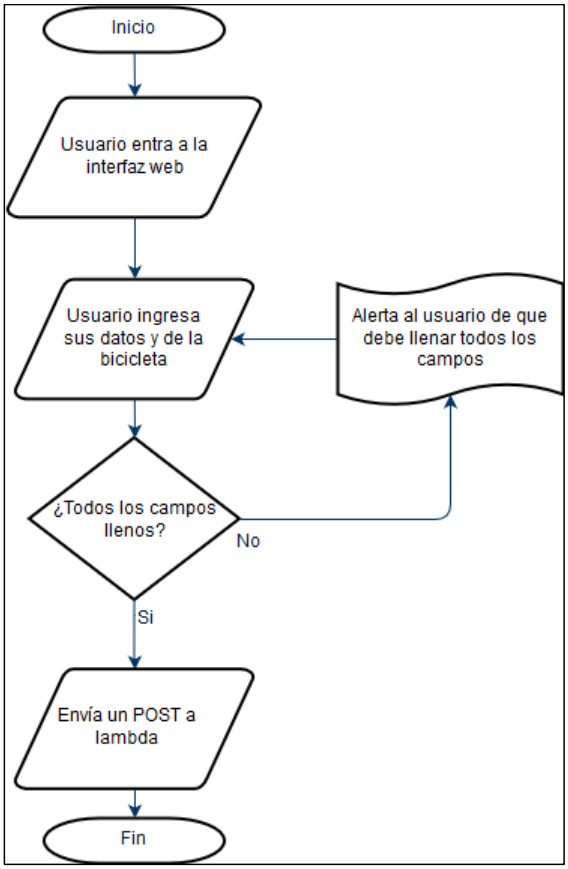


Figura 28. Proceso de registro en la WAPP

Fuente: Autores

Cuando los datos personales del usuario sean recibidos desde el formulario de inscripción, se debe ejecutar un script que debe validar que entre los datos se encuentra la PK[17] recibido, almacenarlo en la tabla de usuarios y notificar al usuario cuando el registro sea exitoso (Figura 28).

[17] Acrónimo de Primary Key

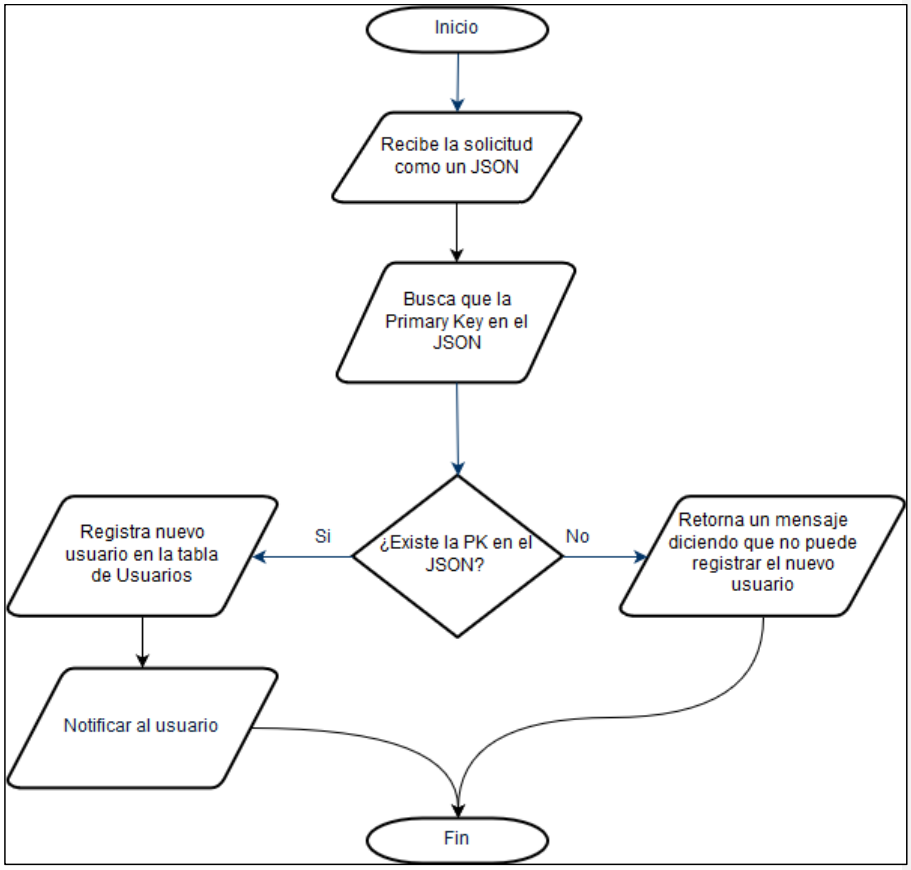


Figura 29. Proceso de registro en la base de datos

Fuente: Autores

Otra función que debe tener la WAPP de valor agregado es la posibilidad de visualizar reportes sobre el uso en tiempo real del cicloparqueadero. Permitir visualizar las marcas de bicicleta que se encuentran registradas en el sistema (Figura 29).

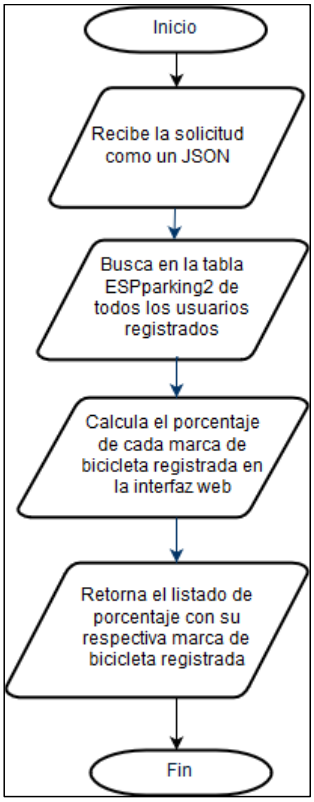


Figura 30. Proceso de generar reporte

Fuente: Autores

5.1.2 IoT Platform Repository

El diseño de esta WAPP debe realizar lecturas y escribir en una base de datos donde deben ir guardados a los registros de ingresos y retiros de bicicletas, como también los datos personales de los usuarios. El diseño del modelo de base de datos considera una relación uno a uno, entre el usuario y el carnet que utiliza para registrar su bicicleta en el sistema (Figura 30).

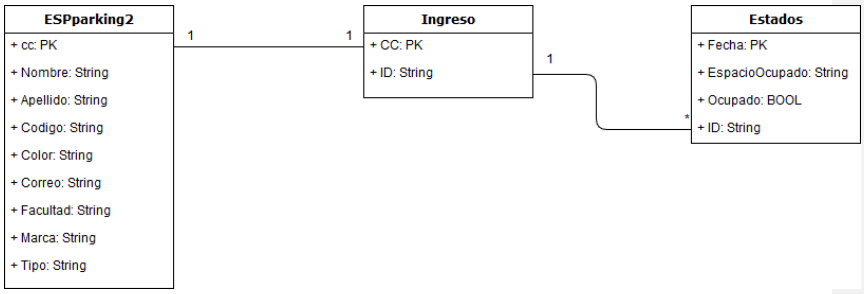


Figura 31. Diseño de modelo de datos

Fuente: Autores

5.1.3 IoT Edge Platform

El diseño del IoT Edge Platform contempla la actividad de los usuarios que ingresan o retiran sus bicicletas. al igual que el almacenamiento de registros en tiempo real, la identificación de los usuarios que están ingresando transportando toda la información hacia el gateway IoT que se encuentre en la nube.

5.2 DESARROLLO

Inicialmente el WAPP se planteó crearlo en un servidor web, alquilando una máquina virtual con un servicio de AWS llamado AWS EC2[18], donde el servidor desde el principio se instalan los softwares necesarios para crear un servicio web para poder controlar todo el frontend y backend de la aplicación, la razón por la cual el desarrollo de la WAPP no siguió ese camino es debido a el modelo de cobro que tiene AWS en el alquiler de máquinas virtuales, donde se cobra una tarifa mensual por tener el servidor encendido y el presupuesto necesario para realizar el desarrollo de la solución era limitado.

Con el fin de encontrar una solución costo-efectiva, se evaluaron otras alternativas técnicas como AWS S3[19], el cual es un servicio de almacenamiento de objetos con disponibilidad de hasta nueve nueves. Además, este servicio tiene la posibilidad de crear páginas web

[18] Acrónimo de Elastic Cloud Compute

[19] Acrónimo de Simple Storage Service

estáticas, para el frontend de la WAPP. El diseño incluye el uso de un bucket de AWS S3 debido a que el método de facturación de S3 es por capacidad de almacenamiento, número de lecturas y escrituras de los archivos alojados al mes, mientras que, en EC2 el cobro era por segundos que estuviera la máquina encendida. En otras palabras, S3 presenta ventajas frente a EC2 en términos económicos.

Albergar el frontend en AWS S3 permitió que el desarrollo del backend se realizara bajo una arquitectura de microservicios. Sabiendo las ventajas y la tendencia que existe por implementar servicios o soluciones en la nube con microservicios AWS permite desarrollar microservicios en un servicio llamado AWS Lambda que es el servicio de Amazon que permite ejecutar código sin necesidad de usar un servidor; la gran ventaja de usar Lambda es que puede usar cualquier otro servicio de AWS para la ejecución de su código, inclusive ejecutar otras funciones lambdas, facilitando la integración con las bases de datos administradas que almacenan la información de los usuarios y del estado actual de la solución IoT.

Algo que caracteriza a las funciones Lambda es que tienen una dirección IP dinámica, es decir estos códigos se ejecutan por eventos que se definen en la creación de dicha función. Es L manera para ejecutar una función Lambda es a través de un servicio llamado AWS API Gateway, donde allí se crea una dirección URL[20] de origen que tiene permitido ingresar la solicitud de tipo HTTP.

Para la elaboración de la interfaz web, se desarrolla el frontend en un archivo con extensión HTML[21], ya que a comparación de los demás lenguajes de programación para la web, este tiene ventajas que favorecieron su uso para la elaboración de este proyecto, una de ellas es que el texto puede ser presentado es de forma estructurada y agradable, de desarrollo ágil y el peso de los archivos es mínimo. Hoy en día no se necesitan grandes conocimientos en diseño de páginas ya que existen editores de páginas web y lo más importante es soportado por todos los exploradores [31].

[20] Del inglés, *Uniform Resource Locator*

[21] Del inglés HyperText Markup Language

De la mano también se encuentra CSS[22], un lenguaje de diseño gráfico para archivos que alser aplicado en bloques HTML da estética a la interfaz web [32].

Para el desarrollo ágil de la Interfaz Web, se utilizó una librería llamada Bootstrap, un conjunto de herramientas de código abierto para el desarrollo de páginas web y diseño de aplicaciones, que contiene botones, gráficas, plantillas, tipografía, formularios y demás elementos basados en CSS y HTML con la facilidad de que la interfaz puede adaptarse automáticamente para ser visualizada en un dispositivo móvil.

Para el desarrollo de las páginas web se desarrolló un script en Javascript que se ejecuta desde el l lado del cliente. Este script se ejecuta al momento de cargar la interfaz web o escuchar eventos que ocurran en los bloques del archivo HTML [33]. Particularmente, en este desarrollo el script se utiliza en la interfaz de inscripción de nuevos usuarios al sistema, justo en el momento en que el usuario da click en el botón al final del formulario, el cual verifica que toda la información sea válida y posteriormente crea el objeto a enviar.

Todos los archivos de extensión HTML, JS[23], CSS e imágenes de la interfaz web, están alojados en un bucket de S3 (Figura 31, 32, 33, 34).

[22] Del inglés Cascading Style Sheets

[23] Acrónimo de Javascript

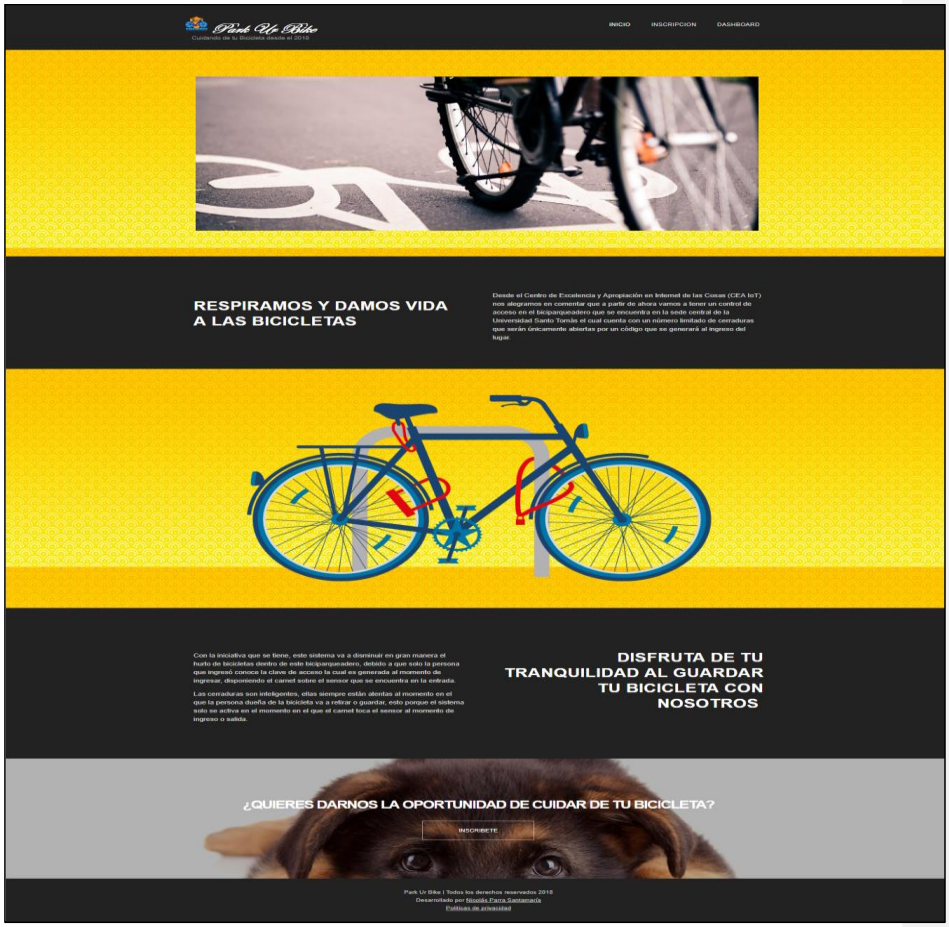


Figura 32. Sección de inicio de la Interfaz Web

https://parkurbike.s3.us-east-2.amazonaws.com/home.html

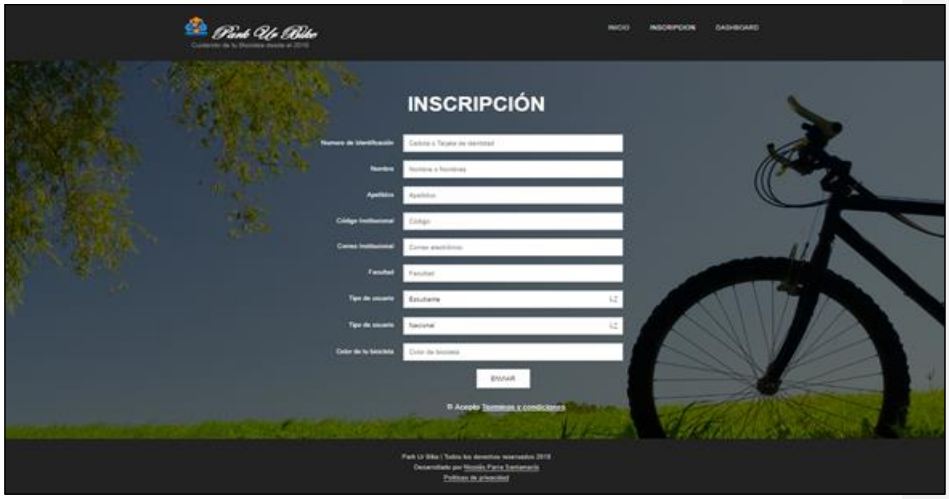


Figura 33. Sección de inscripción de la Interfaz Web

https://parkurbike.s3.us-east-2.amazonaws.com/inscripcion.html

Figura 34. Sección de inscripción de la Interfaz Web

https://parkurbike.s3.us-east-2.amazonaws.com/dashboard.html

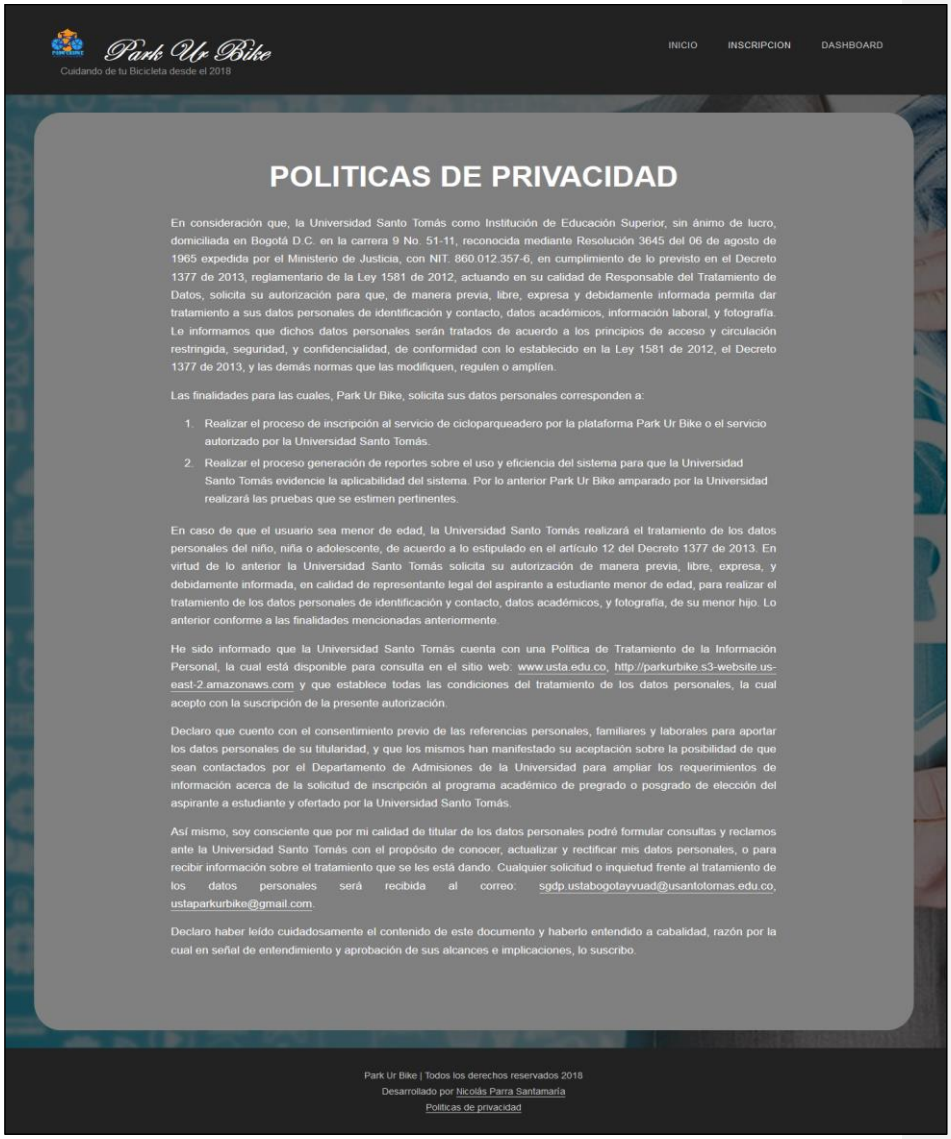


Figura 35. Sección de políticas de privacidad

https://parkurbike.s3.us-east-2.amazonaws.com/Politicas.html

Para el registro de usuarios desde la WAPP se desarrolló una arquitectura de microservicios de 3 funciones Lambda:

- La primera registra nuevos usuarios en la base de datos, específicamente en la tabla ESPparking2 alojada en DynamoDB que contiene la información digitada en el formulario de inscripción.
- La segunda hace una lectura a la tabla donde se encuentran los usuarios registrados, realiza un promedio de marcas de bicicletas registradas y retorna los datos estructurados para que puedan ser graficados en los reportes.
- La tercera registra el estado actual del sistema, almacenando en la base de datos cuando un usuario registra, ingresa o retira su bicicleta en el cicloparqueadero en las tablas de Ingreso y Estados.

Las dos primeras funciones lambdas son ejecutadas a través de AWS API Gateway, la WAPP realiza una solicitud HTTP de tipo POST para registrar un nuevo usuario en la base de datos del servicio o solicitar los datos necesarios para hacer un reporte.

La tercer función lambda funciona como el IoT Gateway propuesto en el diseño y es ejecutada cuando el Módulo Maestro ubicado en la infraestructura registra y hace seguimiento al funcionamiento actual del servicio. Siguiendo la recomendación de Gartner es el IoT Edge Platform, lo hace bajo una estructura definida que la función lambda entiende como escritura o lectura a la base de datos en la tabla que se especifica en la solicitud.

Como gran parte de la infraestructura se encuentra en la nube de AWS, lo que se plantea en el diseño como el IoT Gateway se desarrolla gracias a un servicio de AWS llamado IoT Core, más específicamente en IoT Shadow. Este se encarga de ser un gateway dedicado sólo para dispositivos IoT que se conecten con AWS por el protocolo MQTT[24], donde los topic de sesión por cada dispositivo son generados en el momento en que se crean los dispositivos (things) en IoT Core.

[24] Acrónimo de Message Queue Telemetry Transport

AWS posee un topic genérico que se encuentra en la documentación sobre IoT Shadow, para que el dispositivo que se vaya a conectar a los servicios de AWS pueda autenticarse con IoT Core y con los topic's generados al crear el dispositivo (thing) se conecte directamente con el protocolo MQTT.

Siguiendo el estándar para el desarrollo de soluciones IoT, se implementan bases de datos no relacionales o NoSQL, para ello existe un servicio de bases de datos administradas en AWS, llamado AWS DynamoDB el cual almacenan los datos de nuevos usuarios, el registro de nuevas bicicletas con sus usuarios y el estado en tiempo real. DynamoDB en este caso es el IoT Platform Repository.

El diseño final de la arquitectura en la nube desarrollada en AWS siguiendo la recomendación de Gartner y el diseño planteado previamente se puede apreciar en la siguiente figura (Figura 35).

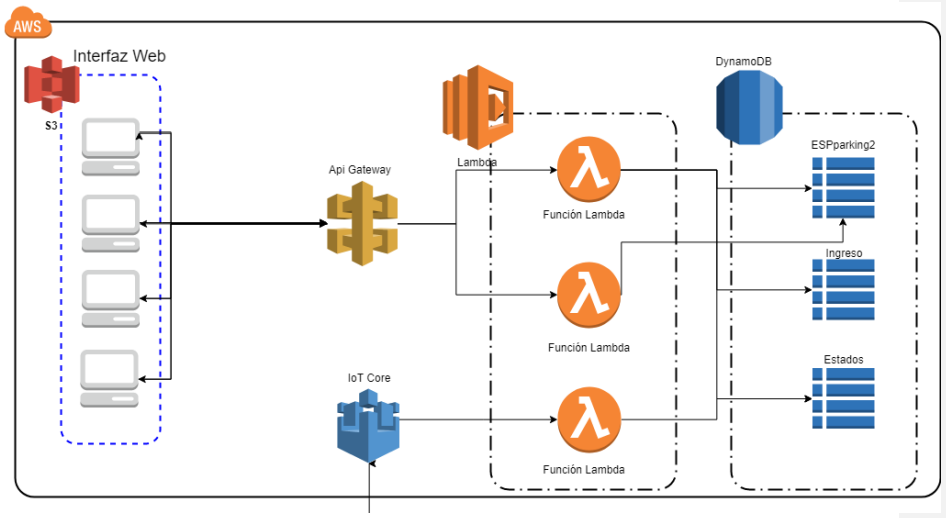


Figura 36. Arquitectura en la nube

Fuente: Autores

5.3 VALIDACIÓN

Para la validación de la arquitectura diseñada y desarrollada para el uso de este sistema se verifica el funcionamiento haciendo una serie de pruebas documentadas en una tabla donde:

- Se enumera la prueba.
- Ingresa el nombre del sector al que se va a realizar.
- Se escribe una descripción muy breve de lo que se va a realizar.
- El resultado esperado.
- El resultado obtenido.
- La prueba fue exitosa o fallida.

Las pruebas realizadas para la validación del funcionamiento se pueden evidenciar en la siguiente tabla.

Tabla 5. Arquitectura en la nube

PRUEBAS PARA LA VALIDACIÓN DEL SISTEMA DE CONTROL DE ACCESO PARKURBIKE					
#	**COMPONENTE**	**DESCRIPCIÓN**	**RESULTADO ESPERADO**	**RESULTADO OBTENIDO Y OBSERVACIONES**	**TIPO DE RESULTADO**
1	WAPP	Enviar un JSON con la información de un usuario que ya se ha registrado en el sistema	Mensaje de que el usuario se ha registrado correctamente	La respuesta del servidor es que el usuario se ha registrado correctamente	Fallido
2	WAPP	Crear un usuario en la tabla ESPparking2 desde lambda	Ver en la tabla ESPparking2 un nuevo usuario con los datos ingresados desde lambda	El nuevo usuario se encuentra almacenado en la tabla ESPparking2	Exitoso

3	WAPP	Enviar un JSON con la información de un usuario de forma incompleta	Retorna un mensaje de error en la escritura en base de datos	El nuevo usuario se encuentra almacenado en la tabla ESPparking2 debido a que la tabla es NoSQL y los datos pueden ser creados de forma dinámica	Fallido
4	IoT Edge Platform	Actualizar el shadow con un json distinto al implementado	Un mensaje diciendo que no se pueden realizar acciones debido a que el JSON no corresponde al planteado	Un mensaje de error diciendo que no existen las llaves correspondientes para continuar	Exitoso
5	WAPP	Realizar una solicitud HTTP al API Gateway sin usar un api key	Un mensaje de status 400 diciendo que no se puede acceder a esa API	Un mensaje con status 400 con mensaje "message forbidden"	Exitoso
6	WAPP	Acceder a la interfaz web por la URL generada por el bucket	Visualizar la interfaz web desde un navegador de un dispositivo móvil y computador personal	Se visualiza de forma exitosa la interfaz web desde ambos dispositivos	Exitoso
7	WAPP	Acceder a la interfaz web a través de la url	Acceso exitoso a la interfaz web	Al acceder a la URL de la interfaz web muestra el contenido	Exitoso
8	WAPP	Registrar un usuario ingresando datos faltantes en el formulario	Un mensaje de alerta que muestre que faltan datos por ingresar	Al ingresar los datos incompletos la interfaz arrojó un mensaje de alerta	Exitoso
9	WAPP	Hacer una solicitud de tipo POST con los datos del usuario por un	StatusCode 200	StatusCode 200 al hacer la solicitud	Exitoso

		JSON a un API			
10	WAPP	Digitar de forma incorrecta un número de cédula de un usuario en el formulario de inscripción	Mensaje de que la cédula deben ser solo números	El formulario permite escribir de forma incorrecta todos los campos que deberían manejar excepciones como el correo, la cédula, el nombre y el apellido	Fallido

Fuente: Autores

La validación para verificar el funcionamiento de la arquitectura en la nube para el sistema de control de acceso del cicloparqueadero inteligente se basó en pruebas de usuario o pruebas de funcionamiento de los componentes implementados para gestionar el cicloparqueadero desde la nube, con esto se logró identificar qué sectores no cumplen con el 100% de la efectividad del sistema. Es por esto que posterior a las pruebas realizadas se logra ver que la arquitectura en la nube se encuentra funcional en un 70% donde el 30%.

Para solucionar el 30% restante se realizaron ajustes y correcciones en la lógica del sistema. Estos cambios se hicieron basados en las descripciones de la validación junto con el resultado esperado por el usuario, actualmente estos cambios se han realizado satisfactoriamente y el sistema hasta el momento cumple con el 100% de éxito en las validaciones realizadas en este proyecto.

6 CONCLUSIONES

Este proyecto presenta el desarrollo y validaciones a nivel de laboratorio de un sistema de control de acceso para el cicloparqueadero de la sede principal de la Universidad Santo Tomás en la ciudad de Bogotá D.C. Este sistema de control de acceso implementa sistemas de hardware, una interfaz web y una arquitectura en la nube enfocada en microservicios. Existe una relación en costos que determinó la razón por la cual este proyecto optó por el uso de implementar arquitecturas de microservicios en la nube y es que cuando un usuario accede a ingresar, retirar o registrar su bicicleta en el sistema de control de acceso, este proceso es ejecutado una sola vez y la nube va a facturar por el uso y la duración de este proceso, es decir que la nube cobrará al sistema por cada usuario que interactúe con el sistema.

Apoyándonos en la implementación de microservicios el sistema cuenta con servicios de valor agregado como: la generación de notificaciones al usuario vía Email del espacio de parqueo en el que quedó guardada su bicicleta, y la visualización de reportes estadísticos sobre el comportamiento de todo el sistema por parte de un administrador a través de la interfaz web.

Para la realización de este piloto de sistema de control de acceso que consta de dos infraestructuras. Una infraestructura física que cuenta con 4 cerraduras conectadas en bus de comunicación un módulo maestro que controla el acceso de bicicletas y un módulo contraseña que valida la contraseña asignada con el espacio asignado. Y una infraestructura en la nube que consta de una interfaz web para el registro de nuevos usuarios y visualización de reportes, una arquitectura basada en microservicios AWS y una base de datos no relacional. Se realizó en un tiempo aproximado de 3 meses desarrollando el proyecto, el sector del cicloparqueadero que más costó tiempo de realizar fue el desarrollo de la arquitectura en la nube debido a que hubo que revisar la documentación de AWS con respecto a los servicios de bases de datos y microservicios. Aprender términos de desarrollo como API’s, serverless, integraciones, requerimientos, Query’s y conocer

protocolos de comunicación para el funcionamiento de los módulos implementados en la infraestructura física.

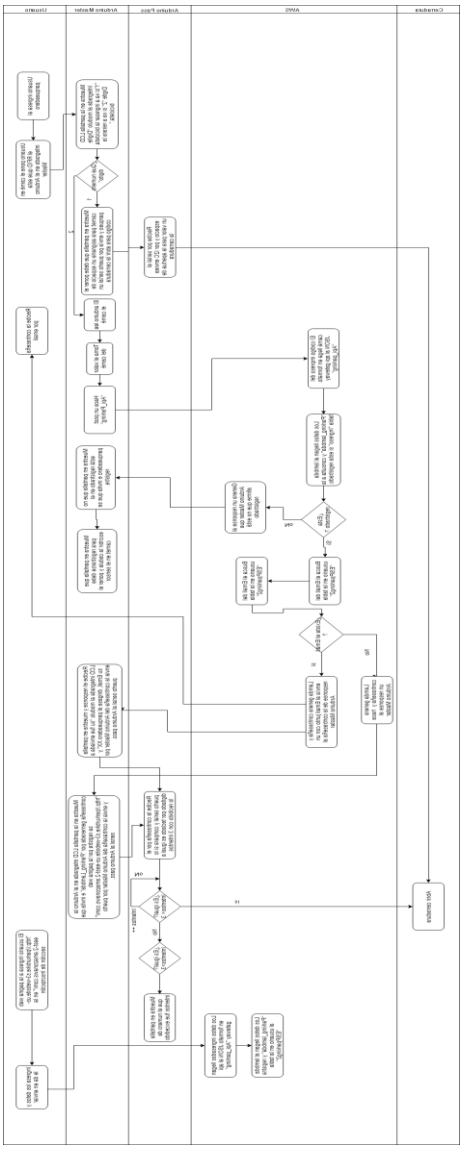
Comentado [1]: experiencia en el proyecto.

6.1 TRABAJOS FUTUROS

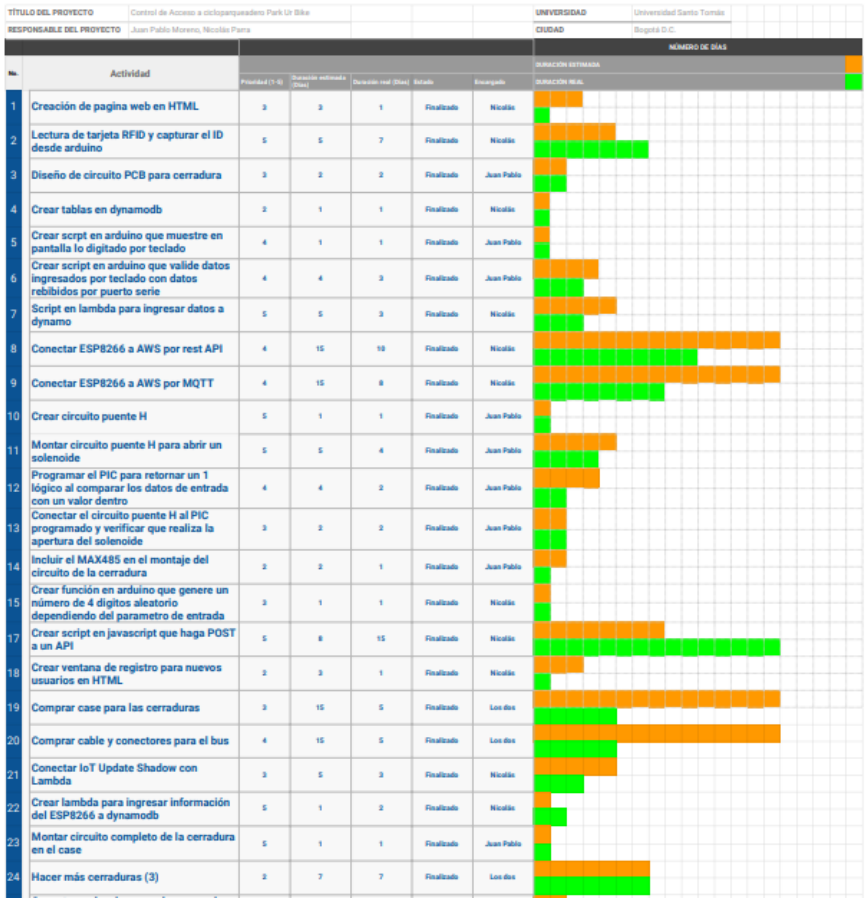
Este proyecto se desarrolló con base a una serie de requerimientos, el cual se fueron cumpliendo cada uno satisfactoriamente con el pasar del tiempo, y a su vez, obteniendo ideas con el fin de mejorar el sistema de control de acceso, dejando como propuesta a futuro la implementación de un módulo utilizando una raspberry, donde dicho módulo cumpla con las funciones del Módulo Maestro y Módulo Contraseña.

Para finalizar, otra idea que puede mejorar el sistema es el uso de la memoria de las tarjetas RFID, se puede aprovechar el uso de la memoria como almacenar información respecto a las características de la bicicleta o también generar unos reportes para identificar el tipo de usuario, es decir, si es estudiante, administrativo o docente.

ANEXO A. DIAGRAMA DE LOS CASOS DE USO



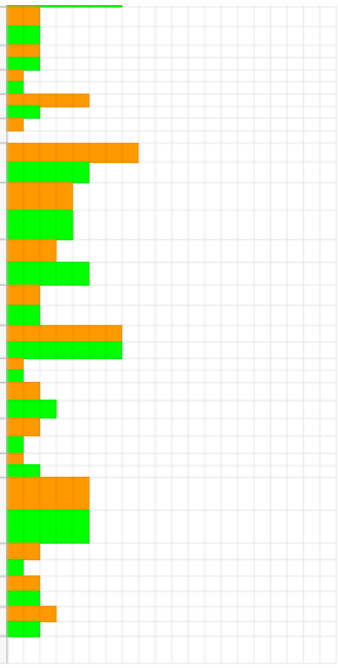
ANEXO B. BACKLOG IMPLEMENTADO PARA EL PROYECTO



TÍTULO DEL PROYECTO	Control de Acceso a ciclopanqueadero Park Ur Bike					UNIVERSIDAD	Universidad Santo Tomás
RESPONSABLE DEL PROYECTO	Juan Pablo Moreno, Nicolás Parra					CIUDAD	Bogotá D.C.

No.	Actividad	Prioridad (1-5)	Duración estimada (Días)	Duración real (Días)	Estado	Encargado
1	Creación de pagina web en HTML	3	3	1	Finalizado	Nicolás
2	Lectura de tarjeta RFID y capturar el ID desde arduino	5	5	7	Finalizado	Nicolás
3	Diseño de circuito PCB para cerradura	3	2	2	Finalizado	Juan Pablo
4	Crear tablas en dynamodb	2	1	1	Finalizado	Nicolás
5	Crear scrpt en arduino que muestre en pantalla lo digitado por teclado	4	1	1	Finalizado	Juan Pablo
6	Crear script en arduino que valide datos ingresados por teclado con datos rebibidos por puerto serie	4	4	3	Finalizado	Juan Pablo
7	Script en lambda para ingresar datos a dynamo	5	5	3	Finalizado	Nicolás
8	Conectar ESP8266 a AWS por rest API	4	15	10	Finalizado	Nicolás
9	Conectar ESP8266 a AWS por MQTT	4	15	8	Finalizado	Nicolás
10	Crear circuito puente H	5	1	1	Finalizado	Juan Pablo
11	Montar circuito puente H para abrir un solenoide	5	5	4	Finalizado	Juan Pablo
12	Programar el PIC para retornar un 1 lógico al comparar los datos de entrada con un valor dentro	4	4	2	Finalizado	Juan Pablo
13	Conectar el circuito puente H al PIC programado y verificar que realiza la apertura del solenoide	3	2	2	Finalizado	Juan Pablo
14	Incluir el MAX485 en el montaje del circuito de la cerradura	2	2	1	Finalizado	Juan Pablo
15	Crear función en arduino que genere un número de 4 digitos aleatorio dependiendo del parametro de entrada	3	1	1	Finalizado	Nicolás
17	Crear script en javascript que haga POST a un API	5	8	15	Finalizado	Nicolás
18	Crear ventana de registro para nuevos usuarios en HTML	2	3	1	Finalizado	Nicolás
19	Comprar case para las cerraduras	3	15	5	Finalizado	Los dos
20	Comprar cable y conectores para el bus	4	15	5	Finalizado	Los dos
21	Conectar IoT Update Shadow con Lambda	3	5	3	Finalizado	Nicolás
22	Crear lambda para ingresar información del ESP8266 a dynamodb	5	1	2	Finalizado	Nicolás
23	Montar circuito completo de la cerradura en el case	5	1	1	Finalizado	Juan Pablo
24	Hacer más cerraduras (3)	2	7	7	Finalizado	Los dos

25	Conectar en bus las cerraduras y probar que hagan apertura cuando reciban el número correspondiente	3	2	2	Finalizado	Juan Pablo	
26	Montar case para el módulo contraseña	3	2	2	Finalizado	Juan Pablo	
27	Conectar y probar módulo contraseña con las cerraduras	5	1	1	Finalizado	Juan Pablo	
28	Montar case para el módulo maestro	2	5	2	Finalizado	Juan Pablo	
29	Conectar arduino con el ESP8266	4	1		En proceso	Nicolás	
30	Crear script en arduino que gestione los espacios de parqueo y genere contraseñas	5	8	5	Finalizado	Nicolás	
31	Crear script en arduino que envíe la información de un usuario ingresando, retirando o registrando su bicicleta por puerto serie a ESP8266	5	4	4	Finalizado	Juan Pablo	
32	Generar JSON y envíar por MQTT a IoT Update Shadow con la información recibida por puerto serie	4	3	5	Finalizado	Nicolás	
33	Conectar case del módulo maestro con el case módulo contraseña y probar apertura de una cerradura	5	2	2	Finalizado	Juan Pablo	
34	Escribir los terminos y condiciones en la pagina web	2	7	7	Finalizado	Nicolás	
35	Montar pagina web a s3	3	1	1	Finalizado	Nicolás	
36	Buscar librerías open source que hagan gráficas	1	2	3	Finalizado	Nicolás	
37	Crear función lambda que genere datos estadisticos	1	2	1	Finalizado	Nicolás	
38	Crear ventana de reportes	1	1	2	Finalizado	Nicolás	
39	Crear función lambda que guarde la información de usuario que se registra desde la ventana de registro y notifica al correo electrónico que se registró exitosamente	1	5	5	Finalizado	Nicolás	
40	Crear script en js que solicite a lambda reportes estadisticos y gráfique	5	2	1	Finalizado	Nicolás	
41	Definir alimentación de los equipos desde la UPS	4	2	2	En proceso	Juan Pablo	
42	Mirar conexión con la UPS el sistema de control de acceso	4	3	2	En proceso	Juan Pablo	
	Número de días propuesto para terminar el proyecto		174	133			



7 REFERENCIAS

[1] Alcaldia Mayor De Bogotá, «bogota,» 25 Julio 2018. [En línea]. Available: http://www.bogota.gov.co/temas-de-ciudad/movilidad/cuantas-personas-se-mueven-en- bicicleta-en-bogota . [Último acceso: 13 junio 2019].

[2] R. Bogotá, «El Espectador,» 06 Agosto 2018. [En línea]. Available: https://www.elespectador.com/noticias/bogota/bogota-ya-cuenta-con-500-kilometros-de-ciclorruta-articulo-804642. [Último acceso: 12 Junio 2019].

[3] T serie H, «Intenational Telecommunications Union (ITU),» 2014.

[4] E. Martínez, «IEBS,» 30 Mayo 2013. [En línea]. Available: https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/. [Último acceso: 12 Junio 2019].

[5] F. N. N. N. N. M. Mohammad Samawat, «Optimization of Wireless Ad-hoc Networks using an adjacent collaborative directional MAC (ACDM) protocol,» International Journal of Computer Applications , 2015.

[6] A. H. K. N. N. F. N. N. Mohd. Saifuzzaman, «ResearchGate,» 10 Mayo 2017. [En línea]. Available: https://www.researchgate.net/publication/317058750_Smart_Security_for_an_Organization_based_on_IoT. [Último acceso: 12 Agosto 2019].

[7] A. Pal, H. Kumar, S. Shailendra y A. Bhattacharyya, «intechopen,» 29 Marzo 2018. [En línea]. Available:https://www.intechopen.com/books/internet-of-things-technology-applications-and-standardization/iot-standardization-the-road-ahead. [Último acceso: 2019 Junio 18].

[8] C. B. K. Hartke, «The Constrained Application Protocol,» Internet Engineering Task Force, Shelby, et al. , 2014.

[9] T. Salman, «Networking Protocols and Standards for,» 30 Noviembre 2015. [En línea]. Available: https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_prot.pdf. [Último acceso: 21 Junio 2019].

[10] J. A. D. T. D. M. Hannes Tschofenig, «Architectural Considerations in Smart Object Networking,» Internet Architecture Board, Tschofenig, et al. , 2015.

[11] M. Bhatia, «Linkedin,» 15 Marzo 2016. [En línea]. Available: https://www.linkedin.com/pulse/internet-things-part-2-mahendra-bhatia/. [Último acceso: 23 Junio 2019].

[12] J. MSV, «Forbes,» ENTERPRISE & CLOUD, 30 Noviembre 2016. [En línea]. Available: https://www.forbes.com/sites/janakirammsv/2016/11/30/artificial-intelligence-and-hybrid-cloud-are-high-on-amazons-agenda/#7ea7b2e073d9. [Último acceso: 10 Julio 2019].

[13] J. MSV, «Forbes,» ENTERPRISE & CLOUD, 12 Mayo 2018. [En línea]. Available: https://www.forbes.com/sites/janakirammsv/2018/05/12/microsoft-starts-to-make-serious-progress-on-the-intelligent-edge-vision/#711a085e2608. [Último acceso: 13 Julio 2019].

[14] J. MSV, «Forbes,» ENTERPRISE & CLOUD, 30 Julio 2018. [En línea]. Available: https://www.forbes.com/sites/janakirammsv/2018/07/30/google-forays-into-edge-computing-through-cloud-iot-edge-and-tpu/#1ba32ab16005. [Último acceso: 13 Julio 2019].

[15] F. D. Sánchez, Cloud Brokering, 2016: Ediciones USTA.

[16] Gartner, “Use the IoT Platform Reference Model to Plan Your IoT Business Solutions”. 17-sep-2016.

[17] D. Angulo-Esguerra, C. Villate-Barrera, W. Giral, H. C. Florez, A. T. Zona-Ortiz and F. Díaz-Sánchez, "Parkurbike: An IoT-based system for bike parking occupation checking," 2017 IEEE Colombian Conference on Communications and Computing

(COLCOM), Cartagena, 2017, pp. 1-5. Available: http://ieeexplore.ieee.org.crai-ustadigital.usantotomas.edu.co/stamp/stamp.jsp?tp=&arnumber=8088201&isnumber=8088182

[18] G. K. Onoriode Uviase, «IoT Architectural Framework: Connection and Integration,» Lancaster, UK, 2018.

[19] M. V. I. B. Ion Lungu, «Database Systems - Present and Future,» Informatica Economica, 2009.

[20] D. M. B. Sharvari Rautmare, «MySQL and NoSQL database comparison for IoT application,» IEEE, 2016.

[21] C. L. P. C. Jing Liu and Yang Xiao, «Authentication and Access Control in the,» 2012.

[22] D. Steele, «Where Is Waze Heading? Into The Internet Of Things,» androidheadlines, 26 Abril 2015. [En línea]. Available: https://www.androidheadlines.com/2015/04/waze-heading-internet-things.html. [Último acceso: 18 Julio 2019].

[23] J. B. Aug, «Waze cofounder tells us how his company's $1 billion sale to Google really went down,» businessinsider, 13 Agosto 2015. [En línea]. Available: https://www.businessinsider.com/how-google-bought-waze-the-inside-story-2015-8. [Último acceso: 19 Julio 2019].

[24] A. Ribeiro, «iot innovator,» 14 Julio 2018. [En línea]. Available: http://iotinnovator.com/esri-waze-provide-near-real-time-data-for-smarter-cities-waze-live-alerts-on-traffic-and-infrastructure-available-in-arcgis-marketplace/. [Último acceso: 2019 Julio 20].

[25] Libelium, «libelium,» 24 Febrero 2014. [En línea]. Available: http://www.libelium.com/smart-water-sensors-to-monitor-water-quality-in-rivers-lakes-and-the-sea/. [Último acceso: 19 Julio 2019].

[26] Grupo Temático sobre Ciudades Inteligentes y Sostenibles, «UIT,» 6 Mayo 2015. [En línea]. Available: https://www.itu.int/es/ITU-T/focusgroups/ssc/Pages/default.aspx. [Último acceso: 20 Julio 2019].

[27] ETSI, «Intelligent Transport Systems (ITS); Communications Architecture,» 2010.

[28] A. Paier, «The end-to-end intelligent transport system (its) concept in the context of the european cooperative its corridor,» de *IEEE MTT-S International Conference*, 2015.

[29] David Angulo-Esguerra, "Documentación piloto de nueve espacios de parqueo del sistema inteligente de cicloparqueadero - ParkURBIke". CEA-IoT nodo USTA, 15-oct-2017.

[30] M. Morales, «teslabem,» 4 Febrero 2017. [En línea]. Available: https://teslabem.com/nivel-intermedio/fundamentos-del-protocolo-i2c-aprende/. [Último acceso: 15 Febrero 2019].

[31] D. P. Valdés, «Maestros del Web,» 02 Noviembre 2007. [En línea]. Available: http://www.maestrosdelweb.com/los-diferentes-lenguajes-de-programacion-para-la-web. [Último acceso: 2 Julio 2019].

[32] Z. L. H. H. Zheng Hao, «IEEE Xplore,» Nanjing, China, 2013.

[33] Con Clase, «HTML 4.01 Specification,» 24 Diciembre 1999. [En línea]. Available: https://www.w3.org/TR/1999/REC-html401-19991224/. [Último acceso: 19 Julio 2019].

[34] M. W. a. S. Thilakawardana, «Initial Analysis of TV White Space,» BBC, UK, 2012.

8 LISTA DE FIGURAS

9 LISTA DE TABLAS