UNIVERSIDAD UTE

FACULTAD DE CIENCIAS DE LA INGENIERÍA E INDUSTRIAS

CARRERA DE INGENIERÍA EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

CREACIÓN DE UN SISTEMA INTEGRADO DE LOCALIZACIÓN PARA EL CAMPUS MATRIZ Y OCCIDENTAL A TRAVÉS DE *BEACONS*, *NFC* E *IOT*

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE

INGENIERO EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

AUTOR:

DANIEL WLADIMIR ALTAMIRANO BORJA

DIRECTOR:

ING. PABLO ANDRÉS SAÁ PORTILLA

QUITO, AGOSTO 2022

FORMULARIO DE REGISTRO BIBLIOGRÁFICO

PROYECTO DE TITULACIÓN

DATOS DE CONTACTO	
CÉDULA DE IDENTIDAD	1714821731
APELLIDOS Y NOMBRES	ALTAMIRANO BORJA DANIEL WLADIMIR
DIRECCIÓN	CARAPUNGO MZ B11 CASA S38
EMAIL	DANIEL.ALTAMIRANO@UTE.EDU.EC
TELÉFONO FIJO	02 2420948
TELÉFONO MOVIL	0995348031

DATOS DE LA OBRA	
TITULO:	CREACIÓN DE UN SISTEMA INTEGRADO DE LOCALIZACIÓN PARA EL CAMPUS MATRIZ Y OCCIDENTAL A TRAVÉS DE *BEACONS*, *NFC* E *IOT*.
AUTOR:	DANIEL WLADIMIR ALTAMIRANO BORJA
FECHA DE ENTREGA DEL PROYECTO DE TITULACION	17 DE AGOSTO DE 2022
DIRETOR DEL PROYECTO DE TITULACION	ING. PABLO ANDRÉS SAÁ PORTILLA
PROGRAMA	**GRADO** ■ **POSTGRADO** □
TITULO POR EL QUE OPTA	INGENIERO EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN
RESUMEN	Este documento presenta un sistema de localización utilizando *Hardware* y *Software*, comprende el uso y desarrollo de aplicaciones web y móviles enfocadas en temas como el *IoT*, b*eacon*s y *Etiquetas NFC*. El sistema permite utilizar *IoT* mediante una placa *Arduino Mega* configurada con un sensor de temperatura resistente a los cambios climáticos. El dispositivo *IoT* permite compartir la variable atmosférica en tiempo real con los usuarios del campus de la Universidad UTE. Los b*eacon*s y *Etiquetas NFC* son introducidos y utilizados mediante el desarrollo de aplicaciones Android. Los b*eacon*s comparten información por el protocolo bluetooth en un rango de 10 metros. Las e*tiquetas NFC* están presentes bajo el protocolo RFID y permiten la escritura y/o lectura de información en su chip de almacenamiento interior. El teléfono Android recupera esta información al detectarlos mediante una de las aplicaciones mencionadas. Además, el sistema cuenta con sitio web que renderiza un mapa de localización sobre el campus

	matriz y campus occidental, mediante herramientas Open-source como Node. js, React. js y Leaflet. js se puede renderizar mapas con un resultado optimo. Este desarrollo permite ubicar edificios, canchas deportivas, eventos y otros lugares disponibles en los diferentes campus. El sitio web funciona en el servidor virtual Digital Ocean, donde el proyecto realizado en Node. js finalmente se comparte bajo la denominación de *ArdUteNfc*. El usuario final accederá a los recursos creados, como a un registro y muro de eventos en el que estudiantes, docentes e invitados no se pierdan de las actividades que se llevan a cabo en la universidad.
PALABRAS CLAVE	*Kanban, Beacon, IoT, MERN-Stack, NFC*
ABSTRACT	This document presents a location system using *Hardware* and *Software*; It includes the use and development of web and mobile applications focused on topics such as *IoT*, b*eacon*s and *NFC tags*. The system allows the use of *IoT* through an *Arduino Mega* board configured with a temperature sensor resistant to climate changes. The *IoT* device allows sharing the atmospheric variable in real time with the users of the UTE campus. *NFC* b*eacon*s and *tags* are introduced and used through the development of *Android* applications. The b*eacon*s share information through the Bluetooth protocol in a range of ten meters. *NFC tags* are present under the *RFID* protocol and allow the writing and/or reading of information on their internal storage chip. The *Android* phone retrieves this information by detecting them using one of the mentioned applications. In addition, the system has a website that renders a location map of the Matriz campus and Occidental campus, using Open-source tools such as *Node. js, React. js and Leaflets. js* maps can be rendered with an optimal result, this development allows locating buildings, sports fields, events and other places available on the different campuses. The website works on the Digital Ocean virtual server, where the project made in Node. js is finally shared under the name of *ArdUteNfc*. The end user will have access to the resources created, such as a registry and a wall of events in which students, teachers and guests do not miss out on the activities carried out at the university.
KEYWORDS	*Kanban, Beacon, IoT, MERN-Stack, NFC*

Se autoriza la publicación de este Proyecto de Titulación en el Repositorio Digital de la Institución.

f:_____

DANIEL WLADIMIR ALTAMIRANO BORJA

ALTAMIRANO BORJA DANIEL WLADIMIR

DECLARACIÓN Y AUTORIZACIÓN

Yo, **DANIEL WLADIMIR ALTAMIRANO BORJA,** CI **1714821731,** autor del Trabajo de Titulación: **CREACIÓN DE UN SISTEMA INTEGRADO DE LOCALIZACIÓN PARA EL CAMPUS MATRIZ Y OCCIDENTAL, A TRAVÉS DE *BEACONS*, *NFC* E *IOT*,** previo a la obtención del título de **INGENIERO EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN** en la Universidad UTE.

1. Declaro tener pleno conocimiento de la obligación que tienen las Instituciones de Educación Superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT, en formato digital una copia del referido Trabajo de Titulación de grado para ser integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública, respetando los derechos de autor.

2. Autorizo a la BIBLIOTECA de la Universidad UTE tener una copia del referido Trabajo de Titulación de grado, con el propósito de generar un Repositorio que democratice la información, respetando las políticas de Propiedad Intelectual vigentes.

Quito, 17 de agosto de 2022

f:_____

ALTAMIRANO BORJA DANIEL WLADIMIR

1714821731

DECLARACIÓN JURAMENTADA DEL AUTOR

Yo **DANIEL WLADIMIR ALTAMIRANO BORJA**, portador(a) de la cédula de identidad N.º **1714281731**, declaro que el trabajo aquí descrito es de mi autoría, que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en ese documento.

La Universidad UTE puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

Quito, 17 de agosto de 2022

f: 

ALTAMIRANO BORJA DANIEL WLADIMIR
1714821731

CERTIFICACIÓN DEL TUTOR

Certifico que el presente trabajo que lleva por título " Creación de un sistema integrado de localización para el campus matriz y occidental a través de *beacons, NFC e IoT* ", que, para aspirar al título de Ingeniero en Informática y Ciencias de la Computación fue desarrollado por Daniel Wladimir Altamirano Borja, bajo mi dirección y supervisión, en la Facultad de Ciencias de la Ingeniería e Industrias; y cumple con las condiciones requeridas por el reglamento de Trabajos de Titulación artículos 19, 27 y 28.

PABLO ANDRES SAA PORTILLA

Firmado digitalmente por PABLO ANDRES SAA PORTILLA
Nombre de reconocimiento (DN): c=EC, sn=SAA PORTILLA, givenName=PABLO ANDRES, serialNumber=IDCEC-1713644258, cn=PABLO ANDRES SAA PORTILLA, 2.5.4.97=TINEC-1713644258001
Fecha: 2022.08.09 12:16:47 -05'00'

f:_____

Ing. Pablo Andrés Saá Portilla
DIRECTOR DE TESIS
C.I. 1713644258

f:_____

Ing. Rodrigo Arturo Proaño Escalante
ASESOR DE TESIS

f:_____

Ing. Segundo Bolívar Jacome Canchig
ASESOR DE TESIS

AGRADECIMIENTO

Muy orgulloso de agradecer a quienes de una u otra manera me brindaron su apoyo y confianza, para llegar al cumplimiento de esta meta tan anhelada que es obtener mi título profesional.

Agradezco:

1. A mi familia: mis padres, y hermanos, que con su amor fueron un apoyo constante en este proceso.

2. A mi tío Álvaro Hernández por su incondicional colaboración en la obtención de mi título profesional.

3. A la UTE al brindarme las facilidades durante los años de estudio y llegar a obtener mi carrera profesional.

4. A mi tutor y director de Tesis, Ing. Pablo Saá, quien con su sapiencia y conocimiento me orientó durante el desarrollo y conclusión del Trabajo de Titulación.

5. A mi amiga y compañera de Universidad, Laura Zambrano, por su ayuda durante la vida estudiantil.

6. A la Consultora Estrategia S.A., en donde realicé mis prácticas.

ALTAMIRANO BORJA DANIEL WLADIMIR

DEDICATORIA

"El sacrificio de día a día tiene su recompensa"

Con amor, agradecimiento y admiración

Para mis padres, hermanos y mi tío**.**

¡ALTAMIRANO BORJA DANIEL WLADIMIR!

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

PÁGINA

ÍNDICE DE FIGURAS

PÁGINA

ÍNDICE DE ANEXOS

PÁGINA

RESUMEN

El presente trabajo propone un sistema de localización basado en *hardware* y *software* que comprende el uso y desarrollo de aplicaciones web y móviles enfocadas en el *IoT*, *beacons* y *etiquetas NFC*. Este sistema permite utilizar *IoT* mediante una placa *Arduino Mega* configurada con un sensor de temperatura resistente a los cambios climáticos. El dispositivo *IoT* permite compartir la variable atmosférica en tiempo real con los usuarios del campus de la Universidad UTE. Los *beacons* y *etiquetas NFC* son introducidos y utilizados mediante el desarrollo de aplicaciones *Android*. Los *beacons* comparten información por el protocolo bluetooth en un rango de diez metros. Las etiquetas *NFC* están presentes bajo el protocolo *RFID* y permiten la escritura y/o lectura de información en su chip de almacenamiento interior. El teléfono Android recupera esta información al detectarlos mediante una de las aplicaciones mencionadas. Además, el sistema cuenta con un sitio web que renderiza un mapa de localización sobre el campus Matriz y campus Occidental de la UTE que, mediante herramientas *Open-source* como *Node. js, React. js y Leaflet. Js,* se puede renderizar mapas con un resultado óptimo. Este desarrollo permitió ubicar edificios, canchas deportivas, eventos y otros lugares disponibles en los dos campus. El sitio web funciona en el servidor virtual *Digital Ocean*, donde el proyecto realizado en *Node. Js* se comparte bajo la denominación de *ArdUteNfc*. Finalmente, el usuario final puede acceder a los recursos creados como a un registro y muro de eventos en el que estudiantes, docentes e invitados no se pierdan de las actividades que se llevan a cabo en la universidad.

Palabras Clave: *Kanban, beacon, IoT, MERN-Stack, NFC.*

ABSTRACT

This document presents a location system using *hardware* and *software*; It includes the use and development of web and mobile applications focused on topics such as *IoT, beacons* and *NFC tags*. The system allows the use of *IoT* through an *Arduino Mega* board configured with a temperature sensor resistant to climate changes. The *IoT* device allows sharing the atmospheric variable in real time with the users of the UTE University campus. *NFC beacons* and *tags* are introduced and used through the development ofAndroid applications. The *beacons* share information through the Bluetooth protocol in a range of ten meters. *NFC tags* are present under the *RFID* protocol and allow the writing and/or reading of information on their internal storage chip. The *Android* phone retrieves this information by detecting them using one of the mentioned applications. In addition, the system has a websitethat renders a location map of the Matríz and Occidental campus in UTE University, using *Open-source* tools such as *Node. js, react. js* and *Leaflets. js* maps can be rendered with an optimal result. This development made it possible to locate buildings, sports fields, events, and other available venues for both campuses. The website runs on the *Digital Ocean* virtual server, where the project was carried out in *Node. Js* is shared under the name of *ArdUteNfc*. Finally, the end user can access the resources created such as abookmark, posters, and events in which students, teachers, and other guestsdo not miss out on the activities carried out at the university.

Keywords: *Kanban*, *beacon*, *IoT*, *MERN stack*, *NFC*.

1. INTRODUCCIÓN

1. INTRODUCCIÓN

El sistema de localización desarrollado aborda temas como el *IoT* en microcontroladores como el *Arduino* y presenta la temperatura junto con sensores como el *DS18B20*, resistente a los cambios climáticos. La localización mediante tarjetas *NFC* permite que cada tarjeta presente esta información en teléfonos con tecnología *NFC*. La tecnología en las tarjetas permite recopilar información mediante *Arduino*, camino viable al *IoT* en campos de seguridad informática.

El sitio web *ArdUteNfc* presenta información personalizada sobre el campus occidental y matriz de la Universidad UTE desarrollado y ejecutándose bajo *Digital Ocean* mediante el servicio de d*ropplets* con el motor *Node. js,* utilizando aplicaciones tanto en *Backend* y *Frontend* en un dominio seguro con *SSL*.

Los *beacons* son pequeños dispositivos basados en tecnología bluetooth de bajo consumo que emiten una señal que identifica de forma única a cada dispositivo. La señal es recibida e interpretada por otros dispositivos, conociendo la distancia a la que se encuentran (Chaparro, 2021).

El *Beacon* para usarse en este trabajo es el modelo *Blue Charm BLE (BC037S),* que emite el formato *iBeacon* mediante el uso de una aplicación del fabricante. Esta se encuentra disponible en la plataforma *Google Play Store* bajo el nombre de *Eddystone*, que permite acceder a configuraciones del *beacon*. Además, emite una señal perceptible para dispositivos con *bluetooth*, que tiene un alcance de 10 m, se lo utiliza casi siempre para notificar anuncios.

Utilicé cuatro b*eacons* para las respectivas pruebas, los cuales emitieron latitud y longitud*,* mediante el desarrollo de aplicaciones en *Android Studio*. En función del *beacon*, la intensidad de señal se categoriza en tres parámetros: inmediata, cercana y lejana. En la Tabla 1 se puede identificar la intensidad inmediata, cercana y lejana, la cual depende del rango del bluetooth que tiene el dispositivo.

Para diferenciar un *beacon* existen los denominados códigos UUID "*Universally Unique Identifier"* por sus siglas en inglés, que en españolsignifica Identificador Único Universal. Este código en muchos casos se puedecambiar en sitios web autorizados con nuevos *UUID* de ser necesario.

Además, otra variable importante es *received signal strengt*h indicator (*RSSI*), en español es el indicador de fuerza de la señal recibida, la cual nos permite conocer el tipo de transmisión de la información, es decir inmediata, pobre o cercana.

La señal de *TX* y *RX* de un *beacon*, se mide en *dBm*, como la intensidad de señal. La variable de latitud y longitud permite conocer en tiempo real la ubicación del *beacon*.

Tabla 1. Datasheet *Blue Charm Beacon BC037S.*

Parámetros *Blue Charm*	**Descripción**
Modo de transmisión:	*iBeacon.*
Nombre del dispositivo:	***pBeacon_n*** (cuando se puede conectar).
	Blue Charm (no se puede conectar).
	El nombre del dispositivo conectable siempre permanecerá como *pBeacon_n.*
***UUID*:**	La pantalla principal de la aplicación muestra 426C7565-4368-6172-6D42.
	Si desea cambiarlo, ingrese un nuevo *UUID* en la pantalla de modificación.
Mayor (*ID2*):	3838
Menor (*ID3*):	4949
Periodo:	1 segundo.
Potencia *TX*:	0 *dBm*.

El *Arduino Mega* 2560 es un microcontrolador. Tiene 54 pines de entrada/salida digital como se detalla en la Tabla 2, un oscilador de cristal de 16 *MHz*, una conexión *USB*, un conector de alimentación y un botón de reinicio (Vizcarra Cavero, 2019).

Tabla 2. *Datasheet Arduino Mega* 2560

Nº	**Descripción**
1.	15 pines pueden ser empleados como salidas digitales.
2.	16 pines analógicas, pueden ser utilizadas con sensores analógicos como temperatura.
3.	*GND*: Pines de tierra.
4.	5 V: Alimentación de 5v para sensores u otros dispositivos utilizados en el proyecto.
5.	3 V: Este pin provee un voltaje de 3.3 voltios para dispositivos que así lo requieran.

La capacidad de conectar el microcontrolador mediante el protocolo *Wi-Fi* implica enviar o recibir datos de Internet, mediante algún protocolo, lo que se traduce en interactuar de forma directa con el usuario o con un bróker automatizado (Valderrama, 2020).

El *ESP-01* soporta la comunicación *I2C*, sus pines serán resumidos en la Tabla 3 y mediante la Figura 1, con tan solo dos pines del microcontrolador se puede comunicar con decenas de sensores *RXD* y *TXD*.

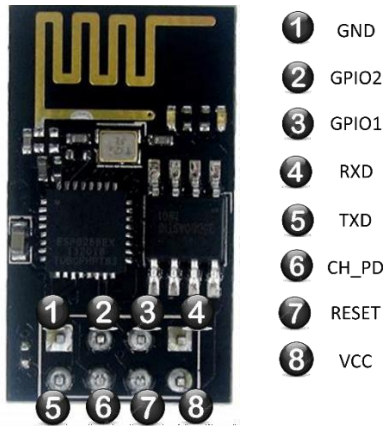

Figura 1. Diagrama del microcontrolador *ESP-01.*

Según (Amaya Fariño, 2020) el *IoT* es una tecnología en la que pequeños dispositivos electrónicos pueden conectarse a Internet permitiendo el desarrollo de nuevas aplicaciones y servicios. La aplicación del *IoT* ha despertado interés sobre todo para el monitoreo de sistemas en tiempo real y ver el resultado mediante el acceso del internet de las cosas.

Tabla 3. *Datasheet* de *ESP*-01.

Nº	**Descripción**
1.	*GND* es la toma de tierra.
2.	*RXD* es el pin donde se reciben los datos del puerto serie.
3.	*TXD* es el pin de transmisión de datos del puerto serie.
4.	*RESET* pin para reiniciar el *ESP*-01.
5.	*VCC*: Funciona a 3.3 V.
6.	La corriente suministrada debe ser mayor que 200 *mA*.

(Setiawan, 2018)

Las tarjetas *NFC* están contenidas en un chip conectado en una antena a modo de bobinado laminado, también llamado *tags* (transmisor y receptor), que se pueden dividir en dos grandes grupos (Sepúlveda, 2018).

Los *tags* activos son más eficientes en ambientes ruidosos, logrando alcanzar mayores distancias. Los *tags* pasivos pueden llegar a ser muy económicos y

pequeños, pero su rango de lectura es muy limitado. Las etiquetas vienen en dos diseños, en forma de llavero y en forma de tarjeta como se observa en la Figura 2. Finalmente, ambos son etiquetas *RFID* que realizan el mismo papel siendo transporte permanente de información.

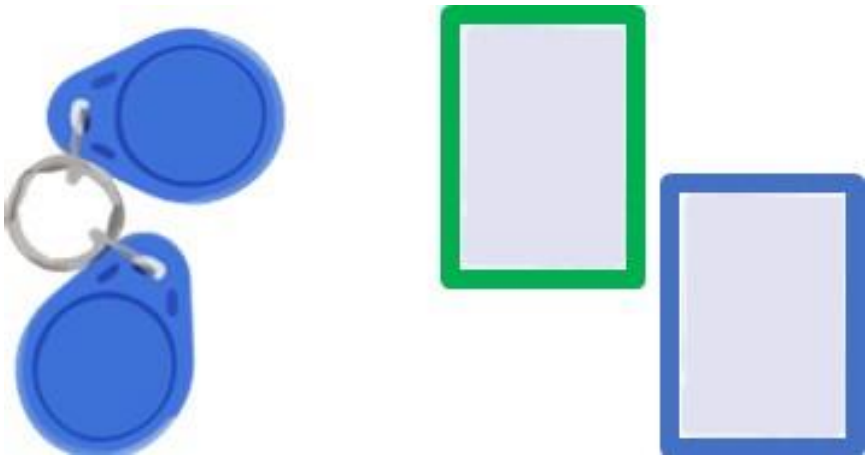

Figura 2. Etiqueta o tag *NFC*.

El microcontrolador *RFID-MFRC522* funciona como lector y grabador de etiquetas radio frequency identification (*RFID*), en español quiere decir identificación por radiofrecuencia. Posee una interfaz serial para la comunicación. Entre el microcontrolador Arduino y la tarjeta MFRC522 se realiza una implementación mediante pines detallados en la Tabla 4. (León Sillo, 2020).

Tabla 4. *Datasheet* de intercomunicación.

Nº	**Módulo MFRC*522***	***Arduino Mega***
1.	*SDA (SS)*	53
2.	*SCK*	52
3.	*MOSI*	51
4.	*MISO*	50
5.	*IRQ*	No conectado
6.	*GND*	*GND*
7.	*RST*	9
8.	3.3 V	3.3 V

(Herrera Llori, 2017)

El módulo utiliza un sistema de modulación y demodulación de 13.56 *MHz*, frecuencia que en la actualidad utiliza la tecnología *NFC*. El módulo se comunica por *SPI*, que significa la interfaz periférica serial, por lo que se puede implementar prototipos con cualquier microcontrolador con interfaz serial, como un *Arduino*.

En la Figura 3 se muestra el sensor de temperatura*, utilizado para la* comunicación a través de solo un cable; además posee un encapsulado de acero inoxidable por lo que tolera la humedad. (Chinga Olmedo, 2021).

Figura 3. Estadísticas técnicas del sensor *DS18B20*.

Este sensor y sus pines resumidos en la Tabla 5, pueden medir una temperatura estimada en el rango de –55° C hasta los 125° C, se puede programar desde nueve bits hasta doce. La librería junto con el *DS18B20* permite realizar lecturas y configuraciones.

Tabla 5. *Datasheet DS18B20*.

Nº	**Módulo *MFRC522***	***Arduino Mega***
1.	Voltaje de alimentación	3 V a 5.5 V
2.	Voltaje de alimentación	*VDD*
3.	Tierra	*GND*
4.	Datos	*DQ*
5.	Rango de temperaturas	-55º C a 125º C
6.	Error (-10º C a 85º C)	±0.5º C

(Siam, 2021)

La comunicación con el microcontrolador se realiza mediante el puerto serie, por lo cual en ningún caso se debe exceder los 3.6 v para los pines *TX* y *RX*. Suministrar energía suficiente desde el *Arduino* supone que en ocasiones puede quedarse corto, sobre todo cuando éste se encuentra en un uso intensivo por parte del protocolo *Wi-Fi* del *ESP-01*.

El *ESP-01* funciona a una tensión de 3.3 V y admite un máximo de 3.6 V. Se recomienda una intensidad mayor de 200 *mA* debido a que cuando está transmitiendo a través de *Wi-Fi* puede alcanzar picos de más de 200 mA (Llamas, 2017).

Un programador *USB*-serie permite programar un *ESP-01* y también alimentarlo. En el mercado existen multitud de adaptadores de este tipo. Lo importante es conocer la corriente que puede suministrar cada uno de ellos. La intensidad que suelen suministrar estos componentes varía entre los 50 mA y los 120 mA.

En cuanto a entornos de programación *Android Studio,* desarrollado y actualizado por *Google*, permite programar en el lenguaje *Java*, *Kotlin* y *C++* (Martínez Lizares, 2020). Para crear aplicaciones de todo tipo, optimizadas para todos los dispositivos disponibles en el mercado actual como teléfonos, tabletas, televisores y automóviles.

Actualmente *Visual Studio Code* es el editor de código gratis más popular del momento, usa el motor *Monaco*, es un editor de código en la web. Para funcionar en múltiples sistemas operativos se usa un complemento conocido como *Electron*, en su entorno de desarrollo *Emmet* es un plugin integrado que sirve para acelerar la escritura *HTML*, es decir, mediante atajos y funciones matemáticas se puede escribir códigos para crear el *backend* y *frontend*.

El *MERN Stack* es una terminología conocida por su desarrollo en el lenguaje *JavaScript*, es decir de inicio a fin, solo se usará este lenguaje. En la Figura 4 se evidencian sus siglas correspondientes a: *MongoDB*, *Express*, *React. js* y *Node. js*.

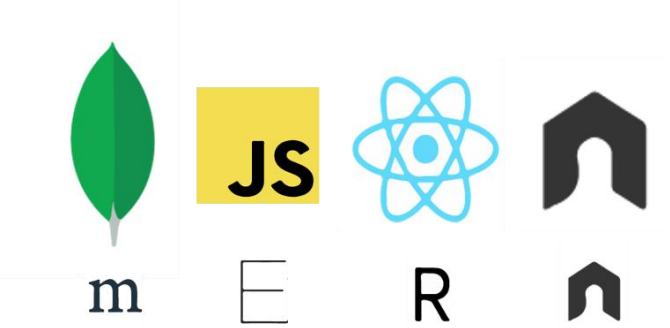

Figura 4. Siglas de los frameworks del stack MERN.

Integrar las cuatro tecnologías permite conocer el *Backend* y el *Frontend*, además de mantener una aplicación escalable, según (Rendón Escudero, 2020). Con el *framework Express*, la programación del *Backend* se reduce favorablemente. La información que es solicitada se procesa y presenta al usuario mediante un CRUD diseñado mediante *MongoDB,* base de datos *NoSQL,* basada en consultas *JavaScript*.

Con ayuda del *CRUD*, las operaciones básicas, en inglés *create*, *read*, *update* y *delete*, podemos crear, leer, actualizar y eliminar información. Este proyecto usa *React. js* para abarcar el *frontend* para facilitar la presentación de los contenidos completando el *MERN stack*.

Esta arquitectura permite producir aplicaciones web más sencillas de realizar, además de hacerlas más ligeras y rápidas, asemejándose a la experiencia que nos daría una aplicación de escritorio (Naranjo Heredia, 2021).

El funcionamiento de *frameworks* como *Ejs* nos ayuda a organizar dinámicamente los archivos *JavaScript* en los recursos estáticos *HTML* y *CSS* que se van creando. A partir de esta cualidad el autor (Vela Galindo, 2019) indica que la principal cualidad de las plantillas *Ejs* es que permiten incrustar el código muy fácilmente en ellas.

El término *REST*, viene a ser cualquier interfaz entre sistemas que usen *HTTP* para obtener datos o generar operaciones en todos los formatos posibles, como *XML* y *JSON*. En la prueba de *REST-API* se utiliza *ThunderClient* que simula ser un cliente y permite realizar peticiones en elementos *HTML*, básicamente se han testeado las rutas *CRUD*, además permite crear situaciones en las que el usuario puede enviar información en el elemento *body* del *HTML* y se establece un tipo de contenido y dato para ser entregado al servidor.

JSX es una extensión de sintaxis similar a *XML* para *ECMAScript* sin semántica definida, desarrollada por *Facebook* y liberada para la expansión entre desarrolladores a nivel mundial.

El propósito de esta especificación implica definir una sintaxis concisa y familiar en las estructuras de árbol con atributos. Según (Jiménez Calderón, 2020), permite que personas menos familiarizadas con *JavaScript* puedan entender y modificar algunas partes del código, lo cual es muy importante cuando se involucran diseñadores o programadores de *CSS* en tu proyecto.

React-Bootstrap se distingue por ser un *framework* que permite la organización de una interfaz web en componentes reutilizables, aportando además una gestión más ágil de los elementos *HTML,* mediante el uso de un *DOM* virtual (Briceño Portilla, 2021). *Bootstrap* funciona bien con archivos estáticos como *HTML* y *CSS*, pero trabajar con *React. js* implica aprender un

Bootstrap distinto, es decir, cada componente ha sido creado desde cero como un verdadero componente de *React. js*, sin dependencias innecesarias como *JQuery.*

Leaflet. js es *un framework que* permite realizar mapas de cualquier parte del mundo, y gracias a *JavaScript se* incorpora interactividad tanto para terminales de escritorio y móviles. En la Figura 5 se visualiza que está diseñado para ser usado de manera instantánea por su sencillez.

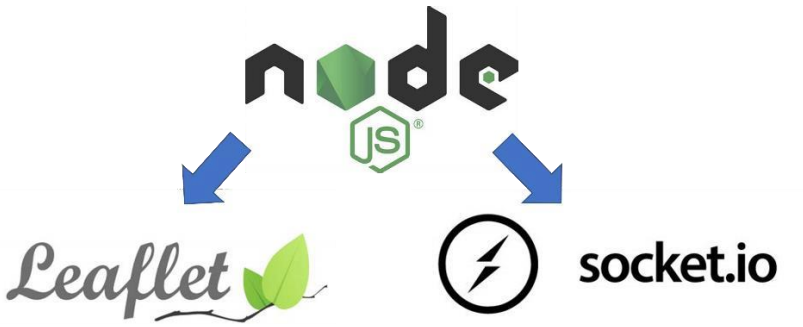

Figura 5. Frameworks del backend con Node .js

En el desarrollo *Nodemon* es un módulo que nos permite reiniciar el código del servidor, aquí se afecta el comportamiento del servidor reiniciando cada vez que un archivo *JavaScript* es modificado. Además, se complementa con el uso del *framework Morgan* para registrar por consola las peticiones que están llegando desde el navegador a las aplicaciones cliente.

AXIOS es un módulo cliente *HTTP* basado en promesas simples para el navegador y cuando es necesario el control de versiones ofrecido por *Git* es lo más utilizado, puesto que se basa en un sistema distribuido. Esto significa que cada usuario contiene una réplica del repositorio, sin depender de tener conexión estable en donde se aloja el código fuente. Esto sumado a la implementación del desarrollo a través de las ramas permite dar rapidez, al poder trabajar con diferentes versiones simultáneamente (Rodriguez Hernandez, 2021).

La plataforma colaborativa de proyectos *GitHub* está destinada a guardar el código de las aplicaciones, dando como funcionalidades adicionales la descarga del proyecto. Colabora en el desarrollo mediante el sistema de issues o correcciones que permite sugerir cambios. Trabaja con el sistema de control de versiones *Git* de forma nativa (Rodriguez Hernandez, 2021).

PM2 es un administrador de procesos de producción para el tiempo de ejecución de *JavaScript*, ayuda a los desarrolladores a administrar y mantener la aplicación en línea las veinte y cuatro horas del día, los siete días de la semana (Samanpour, 2018).

Nginx es un servidor web ligero y de alto rendimiento que también puede ser usado como *Proxy*. Según (Aguilera Blanco, 2021) algunos de los aspectos destacables es que utiliza archivos estáticos, tiene soporte de *HTTP* y *HTTP2* sobre *SSL* y permite usar '*Let's Encrypt*' que proporciona algunos métodos para obtener un certificado *SSL*, estos métodos o las adiciones se denominan autenticación (Ifrah, 2021). Las adiciones se utilizan en el caso de que un certificado debe ser emitido al servidor *Nginx*.

En el *software* privativo se usa Kanbanchi que implementa tableros *Kanban*, utilizando diagramas entre otras funcionalidades en la nube de *Google Workspace*. El *software* es un tablero *Kanban* interactivo, sirve para especificar la pila de entrada del proyecto, visualizar el avance que se ha realizado durante el día, o en la revisión del avance mensual.

Para la prueba del sitio web y entrega de avances se usa el *framework Ngrok* utilizado con la mayoría de los sistemas operativos. Según (Siby, 2020) es la herramienta de prueba más utilizada.

IBM Cloud es un servicio de computación en la nube para empresas y desarrolladores. En la Figura 6 se utiliza en productos IoT. El modelo de negocio de *IBM Cloud* se basa en proporcionar *software* que permita a las grandes corporaciones conectar sus diferentes sistemas, y mejorar su rendimiento de TI (Dávalos-Jiménez, 2019).

Figura 6. Elementos usados en el proyecto IoT.

El dispositivo *IoT* importante para interactuar mediante *Node-RED*, permite la conexión a través de *TCP/IP*, *MODBUS* entre otros (Mateo Castrillon, 2019).

La interconexión es fácil y sencilla. *Digital Ocean* es un alojamiento completo y con soporte para las aplicaciones. Dentro de las opciones, ofrece *droplets*, sistemas operativos y versiones de éstos (Pérez Silva, 2019). Para traducir los nombres de dominio que se implementan en el alojamiento, se utiliza *NameCheap*, base de datos distribuida, usa para recordar direcciones fáciles de usar por las personas reemplazando los números de protocolo *IP* accediendo a recursos en todo Internet.

El objetivo general en el trabajo es desarrollar un sistema para ayudar a la ubicación de personas u objetos mediante el uso de localización, carteles *NFC* y *beacons* que determinan las coordenadas de una persona, u objeto dentro del campo universitario.

Los objetivos específicos del trabajo son los siguientes:

- Realizar un análisis del estado del arte en *IoT*, *beacons*, *NFC* y sus aplicaciones en sistemas de localización *indoor* y *outdoor*.

- Analizar las tecnologías de bases de datos *NoSQL* en la nube, capaces de almacenar la información de *data streamming*, así como la aplicación de técnicas de procesamiento de la data proveniente de los diferentes sensores.

- Desarrollar un prototipo con *arduino* Uno que permita mediante sensores *IoT*, *beacons* y mediante carteles *NFC* capturar la información del campus matriz y campus occidental.

- El usuario, mediante una aplicación móvil / web recibe la información de las actividades que se cumplen en el campus matriz y campus occidental de la Universidad UTE.

2. METODOLOGÍA

2. METODOLOGÍA

Para el desarrollo de este trabajo de titulación se utilizó principalmente la metodología Kanban. El software oficial para la metodología es Kanbanchi de Google y es un sistema de gestión de proyectos en el cual se utilizan tableros como en la Figura 7, para cumplir con los requisitos de análisis, desarrollo de un sistema lógico y físico para finalmente implementar un modelo. Las fases bajo esta metodología son cuatro tableros: ArdUteNfc IoT, Website, Apps y Despliegue.

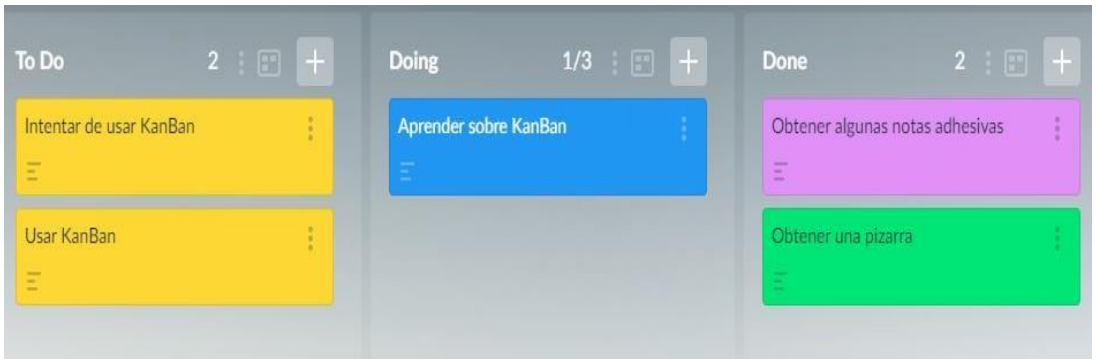

Figura 7. Ejemplo de tablero Kanban.

2.1. METODOLOGÍA KANBAN

Kanban en la planificación de los módulos ayuda a acelerar el proyecto mediante el uso de tableros. En cada tablero se usan tarjetas para definir las fases que ayudan a conocer el estado de un proyecto. Además, esta metodología nos permite visualizar el estado del proyecto en cualquier momento, cumple el rol tanto de enfoque ágil como de herramienta y su objetivo principal es cumplir un conjunto de reglas definidas (Gaete, 2021). La metodología utiliza *WIP*, de las siglas en inglés "*Work in Progress"*, que viene a ser el trabajo en progreso, el cual se controla mediante el flujo de entrega limitando la cantidad de trabajo.

Las tres principales reglas de *Kanban* según (Buñay, 2020) son:

1. Visualizar el flujo de trabajo,
2. determinar el límite del trabajo en curso y
3. controlar el tiempo en completar una actividad.

Visualizar en qué está trabajando cada miembro del equipo, y que todos tengan algo que hacer, siempre teniendo en cuenta las prioridades de cada tarea. Un ejemplo básico de un tablero *Kanban* se demuestra en la Figura 7, la cual consta de 3 columnas que indican el estado de cada actividad. Cada estado tiene un nombre identificativo: *To Do, Doing y Done*, que en español significa: por hacer, haciendo y listo.

A continuación, se hace breve explicación de lo que se realizó en cada una de las fases mediante tableros en esta metodología:

2.1.1. Tablero 1: Fase ArdUteNfc IoT

El desarrollo del sistema IoT, inicia con una recopilación de materiales electrónicos como se puede observar en la Figura 8, se dispuso escoger entre una gama de microcontroladores, sensores, tarjetas, reguladores de voltaje, entre otros que envían y reciben señales analógicas y digitales, es decir información en tiempo real con respuesta afirmativa con la web.

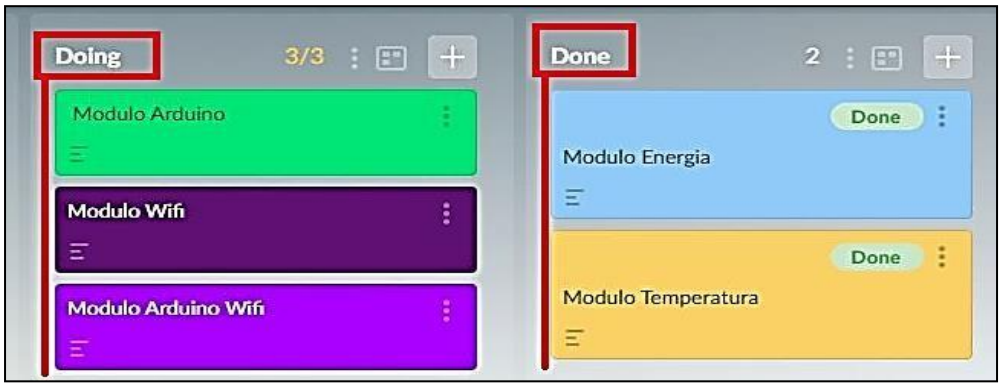

Figura 8. Tablero Kanban para controlar el avance de ArdUteNfc IoT.

2.1.2. Tablero 2: Fase ArdUteNfc Website

El sitio *web ArdUteNfc* fue necesario para presentar los diversos contenidos y aplicaciones referentes a localización e información del campus. En la Figura 9 se observa el desarrollo propuesto para el tablero Website.

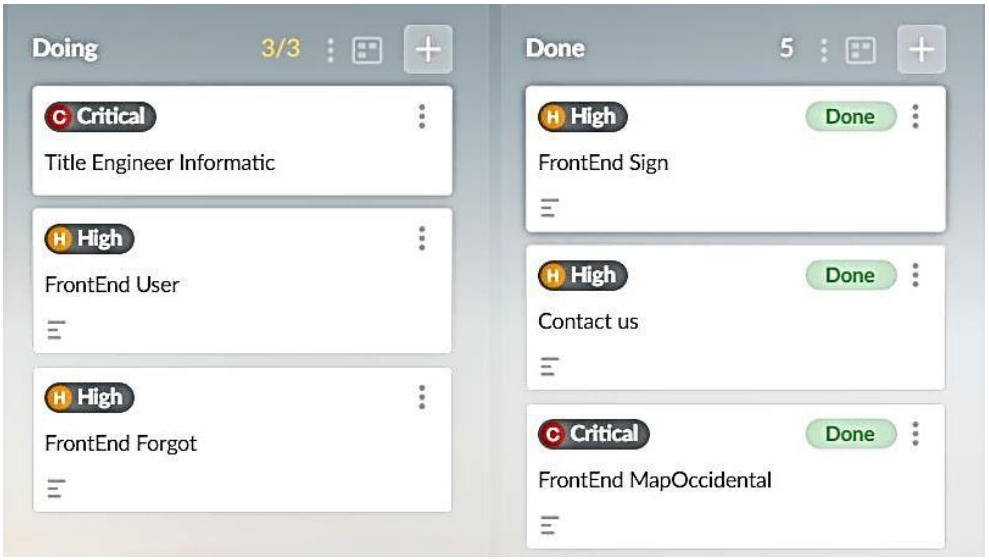

Figura 9. Tablero Kanban para controlar el avance de ArdUteNfc Website.

2.1.3. Tablero 3: Fase ArdUteNfc Apps

Mediante el uso y desarrollo mediante el entorno *Android Studio* podemos llegar a la mayoría de usuarios. En la Figura 10 se visualiza el tablero utilizado para el desarrollo de dos aplicaciones según una *API en común*. Se usó la versión 28 y se actualizo hasta la versión 30 para la compatibilidad.

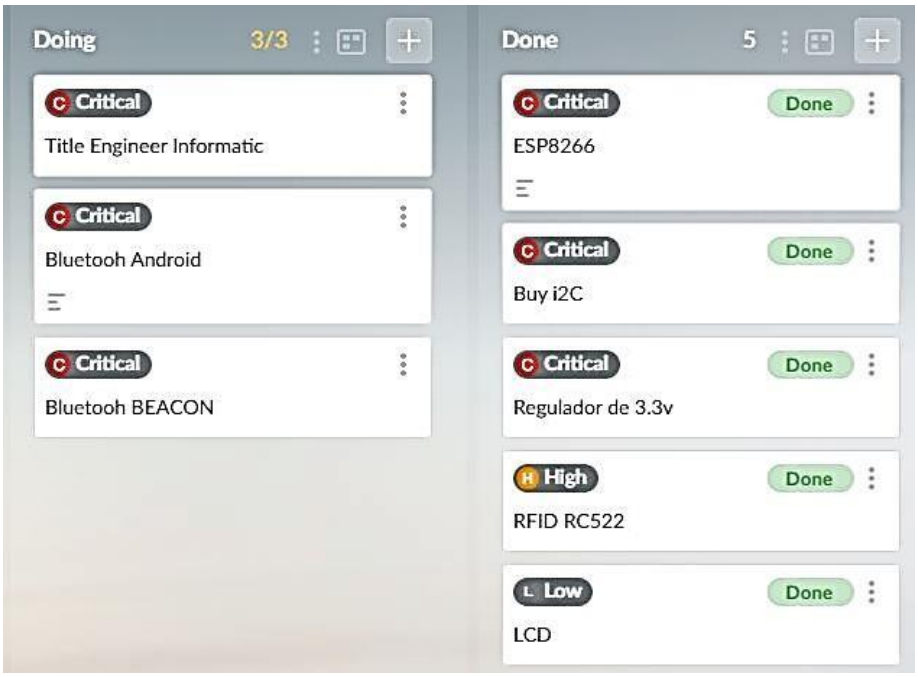

Figura 10. Tablero Kanban para controlar el avance de ArdUteNfc Apps.

2.1.4. Tablero 4: Fase ArdUteNfc *Despliegue*

Esta etapa involucra el desarrollo del tablero *ArdUteNfc Despliegue,* se resume en la Figura 11, fue un ciclo repetitivo y de actualización.

Figura 11. Tablero Kanban para controlar el avance de ArdUteNfc Despliegue.

El tablero *Despliegue* ayuda a finalizar varios procesos como IoT, junto con las dos versiones de *Apps,* NFC y b*eacons*. *ArdUteNfc* Website es el resultado final del sistema de localización propuesto. La tarea final en el tablero es la redacción de documentos, creación de imágenes, entre otros.

3. RESULTADOS Y DISCUSIÓN

3. RESULTADOS Y DISCUSIONES

3.1. Metodología *Kanban*

La metodología Kanban, escogida entre SCRUM y Top Down permitió contar con tiempo para avanzar adaptando, experimentando con el trabajo sin dificultades, y avanzar sin ver el resultado final, pero manteniendo un desarrollo constante del proyecto. No existen metodologías buenas ni malas, cada una posee características únicas que hace que encajen mejor con un proyecto en específico (Remache, 2021).

En la Tabla 6 se puede observar que, entre las metodologías mencionadas, Kanban se escogió por los beneficios iniciales sin material previo y adaptar progresivamente los requisitos del proyecto.

Tabla 6. Comparativa adaptativa con otras metodologías.

	Kanban	**SCRUM**	**TOP DOWN**
Naturaleza	Método adaptativo	Método prescriptivo	Método ciclo
Reuniones	Diario	Diario	Diario
Roles	No se requieren roles predefinidos	Dueño del producto	Dueño del producto
		Maestro Scrum	Maestro Top Down
		Equipo de desarrollo	Equipo de desarrollo
Métrica	Trabajo en progreso	Planificada	Trabajo en progreso

Las empresas de comercio electrónico que seleccioné, fue una de las fases no mencionadas pero esenciales, en resumen, las adquisiciones se realizaron en Ecuador mediante Mercado Libre y en Estados Unidos mediante Amazon.

En la Figura 12 se muestra el avance obtenido a partir del uso de tableros mediante el uso de Kanban, en donde a partir de dos columnas, la primera explica el número y máximo de tareas, la segunda es un listado de los productos finalizados propuestos en el tablero.

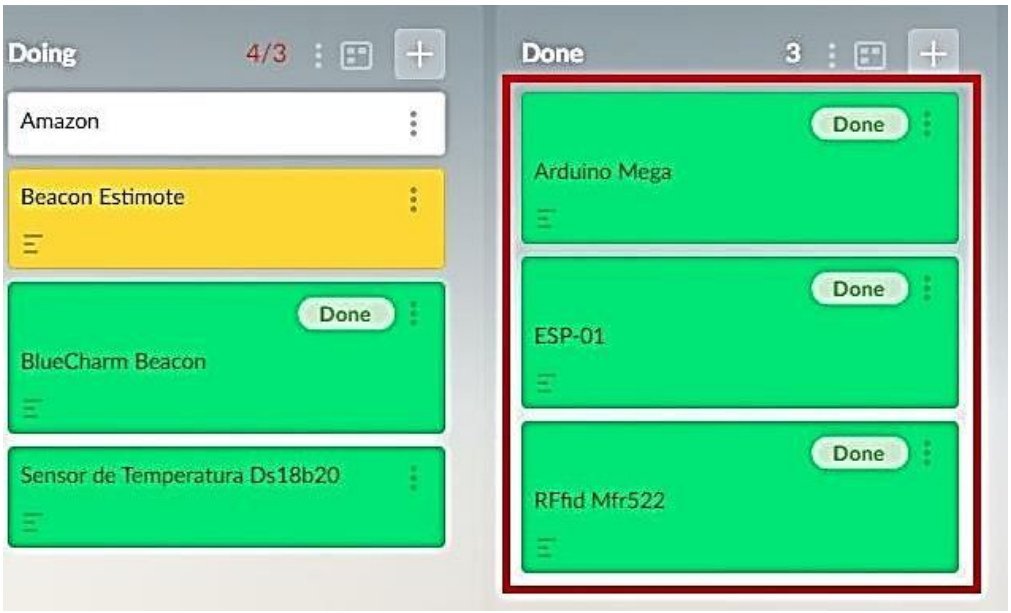

Figura 12. Tablero Kanban para controlar la compra de componentes.

3.1.1. Tablero 1: Fase ArdUteNfc IoT

Análisis de requisitos

El tablero ArdUteNfc IoT parte de la recopilación informacion técnica sobre tarjetas, módulos entre otros, necesarios para la construcción del prototipo IoT. En la Tabla 7 se observan las características básicas analizadas en la elección para el desarrollo del prototipo.

Tabla 7. Comparativa entre microcontroladores disponibles.

Nombre	Arduino Uno	Raspberry Pi	BeagleBone
Modelo	R3	Model B	Rev A5
Precio	$29.95	$35	$89
Procesador	ATMega 328	ARM11	ARM Cortex-A8
velocidad de Reloj	16 Mhz	700Mhz	700Mhz
RAM	2 Kb	256 Mb	256 Mb
i2C	2	1	2
Dev IDE	Arduino	IDLE, Scratch, Squeak	Phyton, Scratch, Squeak

Arduino

Es un microcontrolador presente en la parte lógica y física del circuito, es clave en las fases en las que se conecta sensores, diodos led, tarjetas adicionales como el *RFID-MFRC522 entre otros.* En la Figura 13 se observa el *Arduino* original es decir con código de fábrica identificado con color rojo.

Figura 13. Recursos para interactuar con el *Arduino Mega* 2560.

Lector de etiquetas NFC

Permite la lectura y escritura de *información* en diferentes tipos de datos, se usó la placa *MFRC522* que permite la conexión con el identificador único bajo el estándar *ISO* 1443 para las etiquetas utilizadas.

Se indica que la distancia normada para la comunicación por *NFC* son 10 cm, bajo el estándar *SO/IEC* 1443-3A (Ortiz-Garcés, Diego, & Pozo, 2020). Como se puede observar en la Figura 14, se empleó el *UUID* como la identificación para perfiles de usuario.

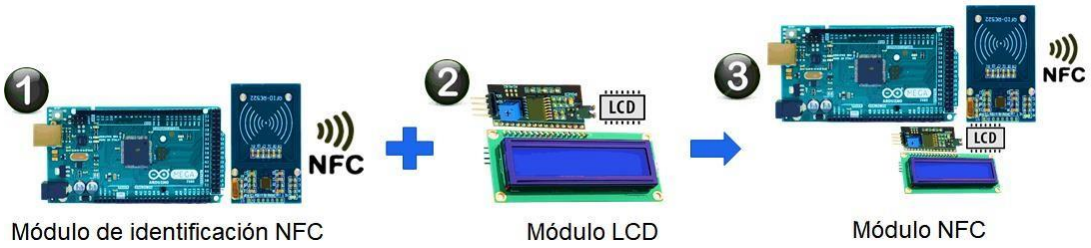

Figura 14. Implementación del módulo *NFC*.

La información que presentan las tarjetas puede ser variada, debido a que se enfocó a ser informativa para el uso de la comunidad de la Universidad, por lo que la tarjeta o *Tag NFC* presentó un texto y/o hipervínculo, además de nueva información bajo el mismo formato, siempre protegida bajo autenticación para evitar la escritura indebida y mantener la integridad de su uso en el campus universitario.

El hipervínculo cargó a los diferentes portales del sitio web oficial de *ArdUteNfc* para obtener más información de la Universidad UTE.

Etiquetas *NFC*

Las etiquetas *NFC* se ubicaron detrás de carteles publicitarios, en donde se ocultó la etiqueta. De esta manera se solicitó escanear la etiqueta con un teléfono con tecnología *NFC* para descubrir información, textos, hipervínculos útiles, ocultos, planificados y presentados previamente con información de la Universidad.

La vulnerabilidad en *NFC* reside en que para escuchar las emisiones de radiofrecuencia entre un dispositivo lector *NFC* y una etiqueta. Estas deben estar lo suficientemente cerca como para detectar los paquetes enviados de información (Ortiz-Garcés, Diego, & Pozo, 2020).

Modulo Wi-Fi

La elección de un módulo wifi, implico estudiar las características más básicas y el stack conveniente para los conocimientos para los cuales me encontraba preparado, en la Tabla 8 se elige el microcontrolador ESP-01, principalmente por el costo y el tamaño del circuito es pequeño y fácil de manipular.

Tabla 8. Comparativa de módulos *Wi-Fi.*

Tecnología	Consumo mA	Costo	Stack
XBEE	50	Alto	Zigbee
ESP-01	200	Bajo	TCP/IP
CC2530	24	Bajo	Zigbee

Programador *ESP*-01

El *ESP-01* tiene una interfaz de información vía serial. La conexión con el entorno de programación disponible en el ordenador fue usada cada vez que se programó. Dispone de dos modos de funcionamiento que se pueden alterar como lo vemos en la Figura 15.

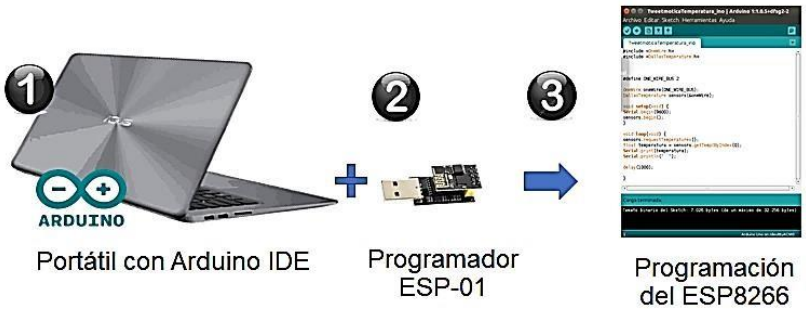

Figura 15. Pasos para implementar un proyecto con ESP-01.

De esta alteración se accedió a usar el *ESP*-01 en modo *UART* o *Flash*. Además, se desarrolló varios prototipos electrónicos, gracias al entorno de programación *Arduino IDE*.

Adaptador *ESP*-01

Esta implementación permite integrar el módulo *ESP*-01 con un circuito de voltaje de 5 V a 3.3 V. Las emisiones de datos pueden presentar errores a nivel de dispositivo, el chip *ESP*-01 necesita una tasa de energía constante de 3.3 V, no estimada para evitar error en la transmisión de los datos.

En la Figura 16. Módulo adaptador para *ESP-01.* se visualiza el módulo regulador de voltaje, un circuito que regula la corriente entrante y saliente manteniéndola estable, de esta manera no se afecta el funcionamiento normal.

Figura 16. Módulo adaptador para *ESP*-01.

Diseño lógico

Lectura visual del Tag NFC

El componente que se muestra en la Figura 17 es una pantalla *Lcd* 16x2, la cual permitió la visualización de datos mediante el protocolo *I2C*. La idea fue transmitir sin necesidad de un computador la informacion en el *Lcd*.

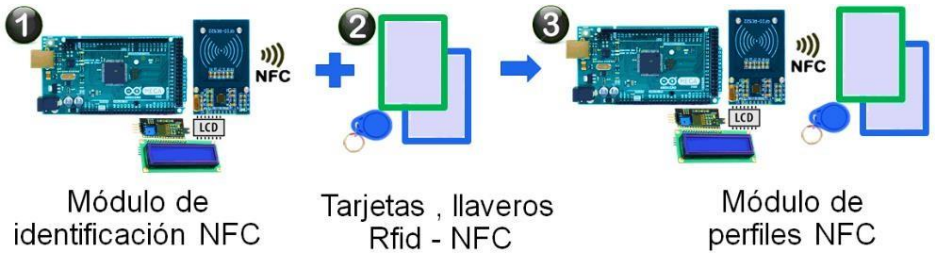

Figura 17. Elementos de una identificación *NFC* mediante *Lcd*.

Leds notificadores

Mediante el encendido y apagado respectivo de un led, indican el comportamiento del sensor gracias a un digito binario constante, cada color tiene instrucciones que se indica en la Tabla 9, el valor 1 para encendido y 0 para apagado. Además, el digito 1 puede indicar paso de información constante.

Tabla 9. Funcionamiento analógico de leds notificadores.

Señal	Descripción
 **Led Azul**	–10 °C a 10 °C.
 **Led Verde**	14 °C a 16 °C
 **Led Rojo**	18 °C a 24 °C.

Regulador de energía

El módulo de energía suministra el voltaje necesario para alimentar el circuito. El funcionamiento del módulo se dividió en voltajes de 3.3 V y 5 V.El voltaje que se administró sin ninguna clase de sistema de energía como intermediario fue de 120 V, por lo cual se consideró usar un adaptador de energía, que convirtió la entrada de llegada de 120 V a 12 V, para iniciar y trabajar con las simulaciones.

Además de este módulo se añadió un circuito de alimentación que suministró 3.3 V y 5 V, necesarios para trabajar con los diferentes módulos y tarjetas usados en el proyecto como se visualiza en la Figura 18.

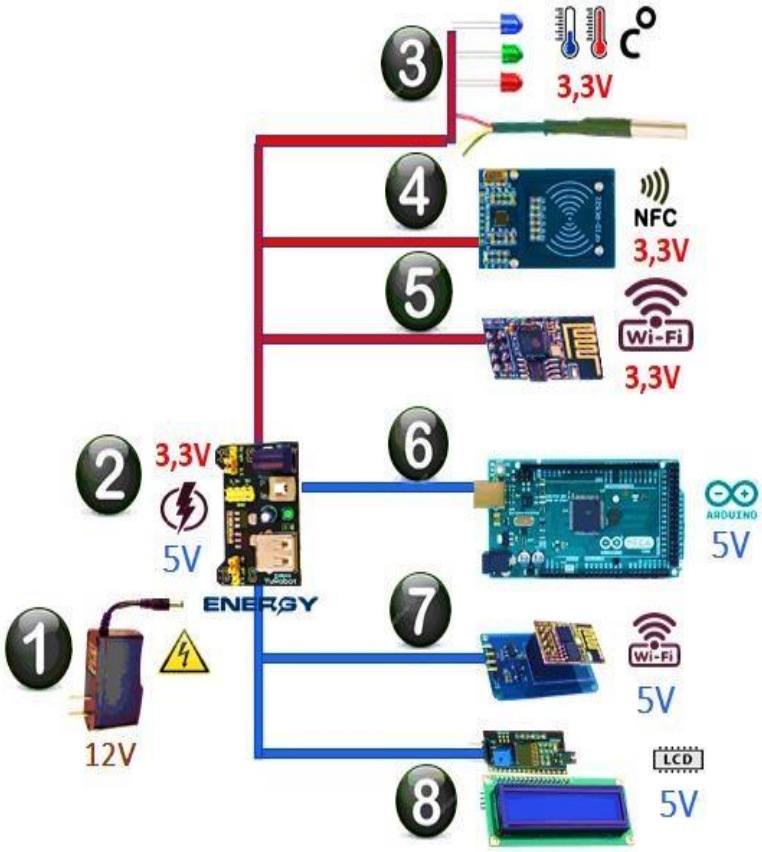

Figura 18. Arquitectura de fuente de alimentación para componentes.

ESP-01 *Wi-Fi*

En la Figura 19 se observa el proceso de los datos captados y enviados por el microcontrolador *ESP*-01 mediante el protocolo *Wi-Fi* a la nube. Finalmente, estos datos fueron entregados y visualizados en navegadores.

Figura 19. Elementos del Módulo *Wi-Fi.*

Modos de programa *ESP*-01

El programador para el *ESP*-01 permite dotar de ejemplos de *Wi-Fi*, se utilizaron programas como *"Firmata", "AccessPoint", "WifiServer",* entre otros que ayudaron a enviar los datos al ordenador y más adelante a la nube. La Figura 20 resume la conexión del microcontrolador al nuevo *Software*.

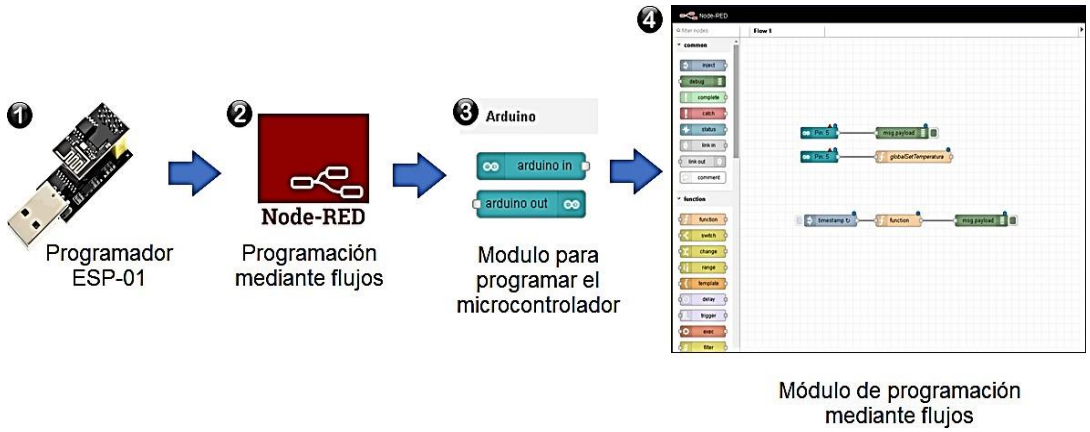

Figura 20. elementos de comunicación local y en la nube.

Firmata

Es un programa que estableció la comunicación con la interfaz *Node-RED*, en la Figura 21 se observan los pasos para transformar microcontroladores en dispositivos *IoT*, además de la comodidad de una programación de *Backend* sencilla.

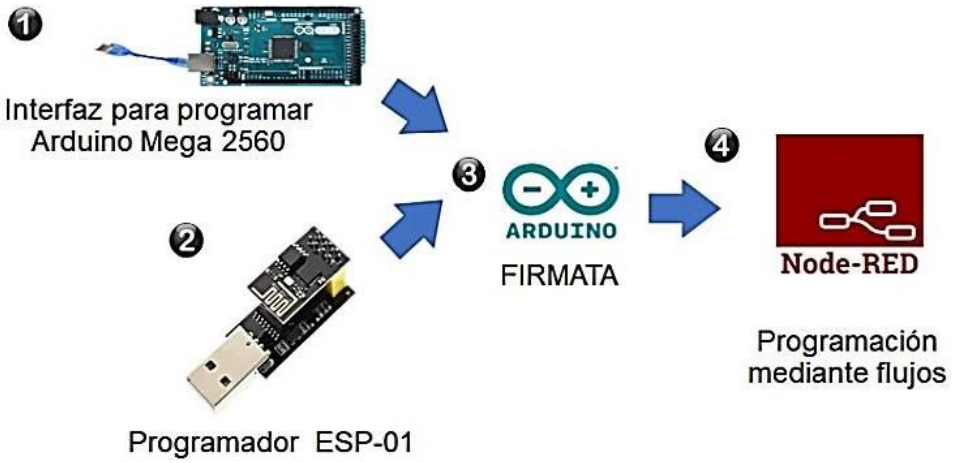

Figura 21. Pasos para una programación en *Node-RED*.

Diseño físico

Arduino *Wi-Fi*

Permitió tanto al *Arduino* como al *ESP*-01 publicar en servicios de la nube. La Figura 22 representa la conexión entre los pines *TX* y *RX* los cuales trasmitieron la información en ambos microcontroladores. De esta manera el *Arduino* logró enviar los datos que generó el sensor de temperatura.

Figura 22. Conexión de pines RX y *TX* creando un el módulo de internet.

Sensor de temperatura con *Arduino*

Es necesario colocar en el sensor *DS18B20* una resistencia de 4.7 K Ω, para proteger del voltaje. Entre los componentes para programar es necesario el portátil; sensor de temperatura; microcontrolador *Arduino*; leds Indicadores de color rojo, verde y azul; además del módulo *ESP-01* para trasmitir la información mediante Internet.

En la **Figura 23** se visualiza el *Arduino* para compartir la información del sensor por diferentes medios. Además, el *LCD* permitió visualizar los datos en tiempo real a los usuarios.

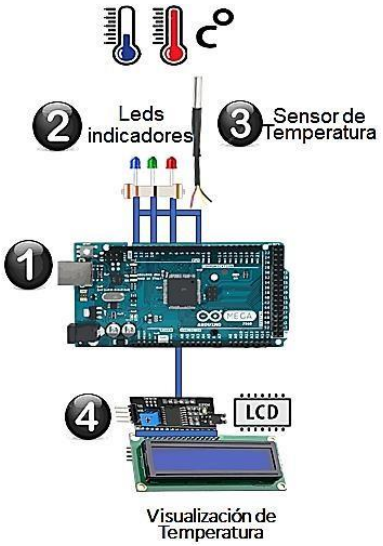

Figura 23. Elementos del módulo temperatura.

Implementación

Librerías del sensor de temperatura

El sensor de temperatura admite temperaturas tan bajas como altas, además se encuentra cubierto en esta versión con acero inoxidable. El sensor *DS18B20*, en la Figura 24 se presenta dos tipos de lectura según la librería que se use, en el caso de esta implementación se usa 1-*Wire* y *DallasTemperature*. La temperatura se presentó estable a los cambios, por lo que es indispensable después de algunas modificaciones y ajustes funcione por horas.

El *Arduino* necesitó estar siempre conectado y funcionando mediante un puerto USB del computador para presentar la información y hacerla disponible en el servicio en nube.

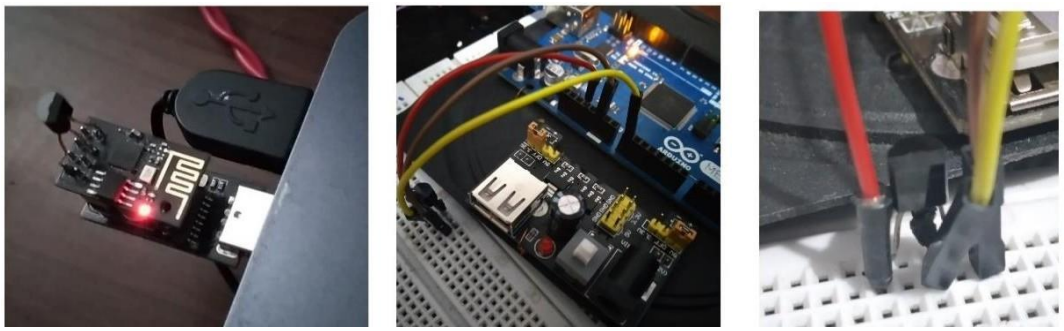

Figura 24. Pruebas con componentes comprados sobre temperatura.

Librería *1-Wire*

La Librería *1-Wire* funciona bien en la presentación de datos por un largo periodo de tiempo, es usada en este proyecto, porque la devolución de datos era exacta y para un ambiente de funcionamiento por horas resultó de utilidad.

La Figura 25 resume las líneas de código sobre la implementación de la librería y el código de lectura para temperatura usado en esta versión.

Temp01.ino §

```
//Librerias
#include <OneWire.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
//Instancia de Objeto Principal de las librerias
OneWire ow(10);                    //Temp Cable de Datos
```

Figura 25. Uso de la librería *1-Wire* en el *sketch* final.

Puerto serial del Arduino

Se puede programar para emitir por el puerto serial como en la Figura 26. Una vez que este puerto es leído por el ordenador se puede acceder a los datos continuos del *microcontrolador*. El Arduino transmite los datos por largos períodos. El *ESP-01* también lo hace, pero durante cortos períodos y además necesita de monitoreo debido a que suele dejar de funcionar el programa precargado.

Figura 26. Lectura del sensor de temperatura mediante el puerto serial.

El propósito de esta información constante en un servidor es interactuar además con el conocido frontend para presentar los datos de temperatura en tiempo real.

Perfiles *RFID-NFC*

Este módulo permite observar cada uno de los *UID*, identificadores únicos universales de cada etiqueta NFC. La utilidad de este módulo es conocer el código único de la etiqueta y presentar un enlace *URL*. En la Figura 27 se detalla la solución asignada mediante el microcontrolador Arduino.

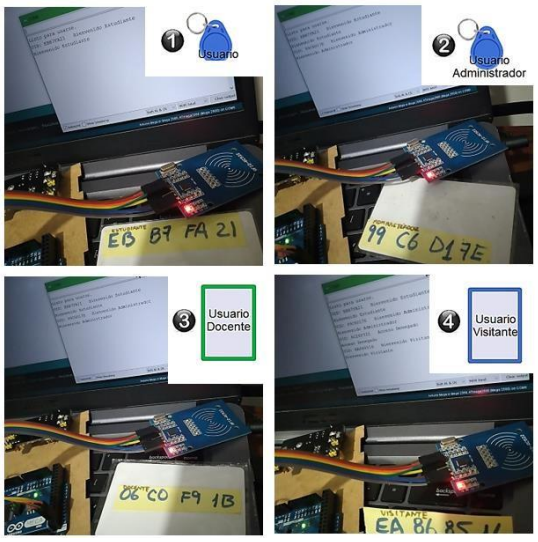

Figura 27. Posible solución encontrada para los datos del módulo RFID.

Fuente de alimentación

La mejor opción para alimentar un *ESP-01* es una fuente de alimentación que suministre 3.3 V y/o 5 V, lo que es una ventaja, ya que muchos de los sensores y componentes que se utilizaron necesitan 5 V.

La fuente de alimentación distribuida por *Mgsystem* cumple con las condiciones necesarias y está disponible la Datasheet en Internet para el uso de la comunidad. Además, cada módulo o tarjeta añadido dispone de un funcionamiento único que divide el voltaje necesario como prosigue:

- **12 V:** En general es el voltaje que permite que componentes de *Hardware* potentes arranquen y funcionen con normalidad.

  Los requisitos mínimos para el funcionamiento de cada componente los especifica el fabricante y se debe suministrar el mismo voltaje parano quemarlos o tener un mal funcionamiento. Resumido en la Figura 28, Tabla 10, que en diferentes ocasiones resultó óptimo.

- **5 V:** El puerto USB de un ordenador proporciona 5 V, este voltaje suministra energía suficiente para el adaptador *ESP*-01, *Arduino* y *Lcd i2c*.

- **3.3 V:** Se suministra para dispositivos que cargan información en una memoria, por ejemplo, el módulo *MFRC522*, *leds* indicadores, *DS18B20* y el *ESP*-01 solo.

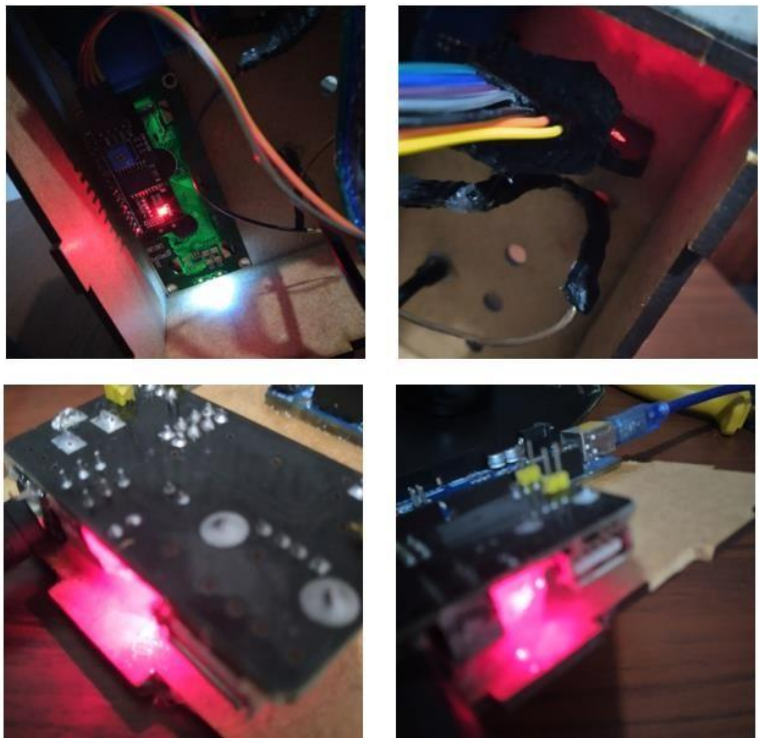

Figura 28. Prueba de componentes con voltaje correcto.

Tabla 10. Módulo de energía para componentes de *ArdUteNfc IoT*.

Energía 3.3v:	**Leds indicadores**
	Módulo temperatura
	Módulo *NFC*
	Módulo *Wi-Fi*
Energía 5v:	Módulo *Arduino*
	Módulo *Wi-Fi*
	Módulo *Lcd*

Conexión microcontroladores y software *Node-RED*

La conexión del Arduino debe pasar antes por un proceso de cambio de entorno de programación, es decir, desde el *IDE Arduino* se carga un programa denominado *Firmata*. Una vez cargado en el microcontrolador éste, se puede establecer comunicación usando *Node-RED*.

El sitio web del microcontrolador se aloja en una dirección *IP protegida,* a la cual se puede acceder desde un teléfono inteligente, esto debido a que no dispone de un dominio al ser un circuito básico.

Para cargar el programa a través del puerto serie como en la Figura 29, se utiliza los pines *RX* y *TX* para transmitir y/o recibir los datos del microcontrolador, donde se almacenará el programa.

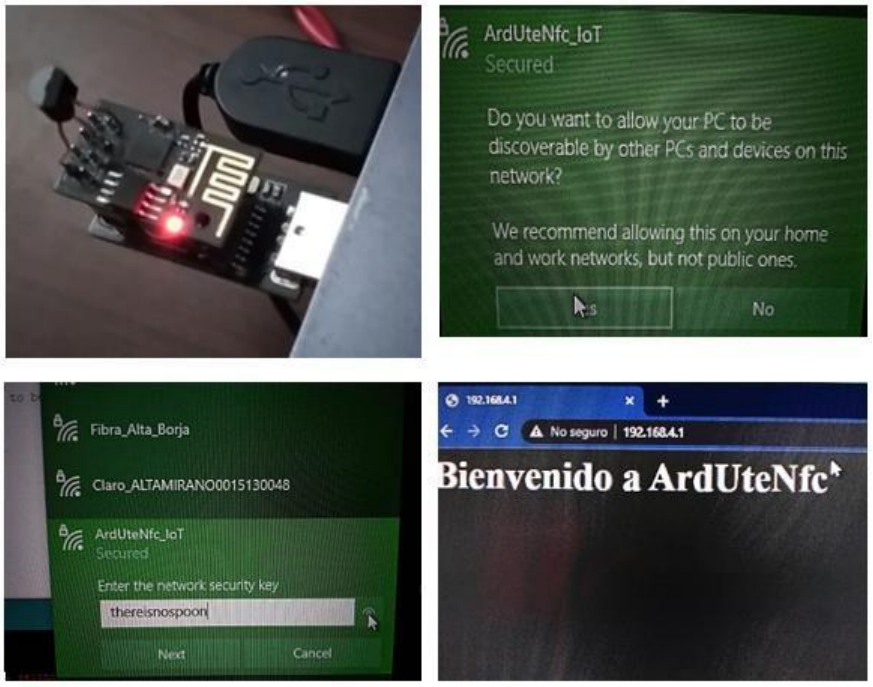

Figura 29. Acceso a la red mediante *ESP-01*.

Existen dos modos de operación tratados en la Tabla 11, que muestra ambos modos y los pines se asignan a los estados *HIGH* y *LOW* para controlarlo. El lenguaje con el que se va a programar es el nativo de *Arduino*.

Tabla 11. Conexión de pines para modos de programa en *ESP*-01.

Modos de programa	***GPIO0***	***GPIO2***
Modo *UART* (carga programa)	*LOW*	*HIGH* (desconectado)
Modo *Flash* (ejecuta programa)	*HIGH* (desconectado)	*HIGH* (desconectado)

3.1.2. Tablero 2: Fase ArdUteNfc Website

Análisis de requisitos

El tablero ArdUteNfc Website usado para la construcción del sitio w*eb* implementó s*oftware* como Node .js, Express, Websockets, Leaflet .js, entre otros. En la Tabla 12 se observa un estudio previo sobre las tecnologías que se pueden utilizar, donde se analizó la curva de aprendizaje, rendimiento y administrador de paquetes disponible en cada uno.

Tabla 12. Comparativa de lenguajes de programación backend.

	PHP	**Node.js**	**ASP.NET**	**Django**
Lenguaje	PHP	JavaScript	C#, F#, Visual Basic.	Python
Uso	Open-Source, Back-end, Web Apps.	Open-Source, Web API, Web Apps, Frameworks, Scripts.	Microservicios, Docker, IA, Servicios Cloud.	Manejo Complejo de base de datos, Web Apps, entre otros.
IDE	Cualquier editor	Cualquier editor	Visual Studio	Python IDE
Curva de Aprendizaje	Media	Media	Alta	Baja
Rendimiento	Lento	Liviano	Pesado	Bajo
Administrador de paquetes	Composer	Npm	Nugget	Pip

JSX

Permite transformar el código usado en el diseño del f*rontend* y se usa comúnmente para transformar el código *HTML*, *CSS* y *JavaScript* al formato *JSX*. Tanto el servidor *backend* y f*rontend* se basan en el mismo lenguaje, esto genera una ventaja en el desarrollo. En la Figura 30 se visualiza un elemento *JSX* que reemplaza al conocido *href*.

```
<Link
  to={{
    pathname:
      "https://hh9fgm.internetofthings.ibmcloud.com/dashboard/
  }}
  target="_blank"
>T of Campus
</Link>
```

Figura 30. Elemento *link* para direccionar mediante *React. js*.

Los estudiantes y docentes que utilicen el sitio web *ArdUteNfc* serán parte de la base de datos y una vez registrados podrán realizar actividades básicas en la publicación de eventos que se desarrollarán en el campus.

EJS

Ejs. js y el framework Express para el b*ackend* alojan el servicio de mapas de cada campus de la universidad. Los recursos de mapas, se implementan mediante *Leaflet*. *js* en archivos dedicados a cada campus.

Socket.io

Esencial para implementar *Websockets* en *Node .js* y resolver la interactividad que el usuario busca para la localización en el ambiente universitario.

Diseño lógico

El sitio web al ser considerado estático alojará la base de datos y aplicaciones desarrolladas mediante un sencillo hipervínculo, siendo un enlace directo para su descarga manual, contacto y uso oficial.

Análisis base de datos

La base de datos escogida es *MongoDB*, debido a la escalabilidad deseada, es la encargada de almacenar la información del sitio web, se resume en la Figura 31, maneja *JSON* para las consultas con respecto a los datos *NoSQL*.

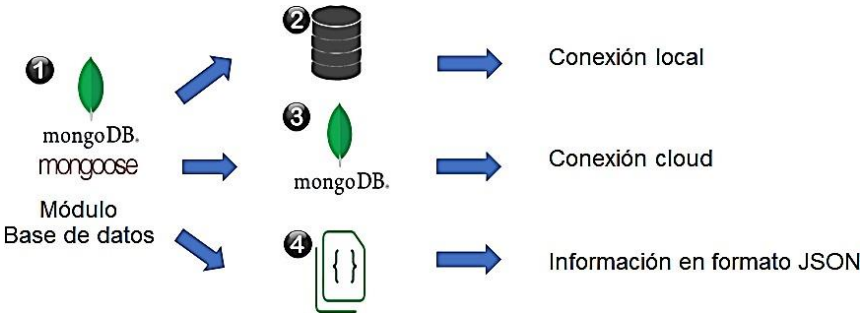

Figura 31. Arquitectura de la base datos.

La base de datos *MongoDB* funciona de acuerdo con una *REST-API*, en la Figura 32 se observa la definición que se realizó mediante peticiones de diferentes tipos, entre las usadas para cumplir el *CRUD* se encuentran *GET*, *POST*, *PUT* y *DELETE*.

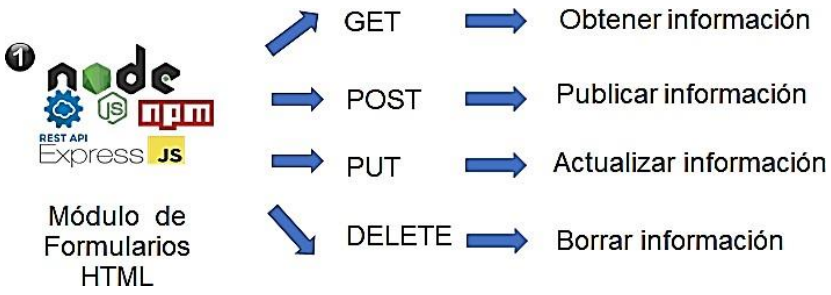

Figura 32. Métodos que conforman el *CRUD*.

Ambos sitios web proponen desarrollar mejor la programación, por lo que en la consola del navegador se puede navegar y conocer detalles de las funcionalidades experimentadas y documentadas.

Diseño físico

Configuraciones con *Express*

Se configura el proyecto para programarlo localmente. Además, se configura que la salida que tendrá por defecto se realice por el puerto asignado del servidor en la nube. Estas configuraciones son esenciales e importantes para desplegar la aplicación web.

Intercambios o *Middlewares*

Los Middlewares son funciones que se ejecutan antes de que lleguen a las rutas, por ejemplo, en la Figura 33 se usa el módulo *Morgan* para controlar las peticiones de los usuarios usando *Ngrok* y *Nginx*.

Figura 33. Ejemplo de notificación con *Morgan*.

La notificación indica que una aplicación cliente se solicita mediante una petición *Get* para obtener un recurso estático como ruta dentro del servidor *ArdUteNfc*. La respuesta es un código 404, debido a que no existe nada en esa ruta. El tiempo de respuesta toma 3.588 milisegundos y el peso de la respuesta es 139 *bytes*.

Además, en la Figura 34 se visualiza el f*ramework Express* que incorpora una función *JSON*, que permite escribir y responder en código *JSON a partir de rutas HTML con informacion precargada de funcionalidad o datos*.

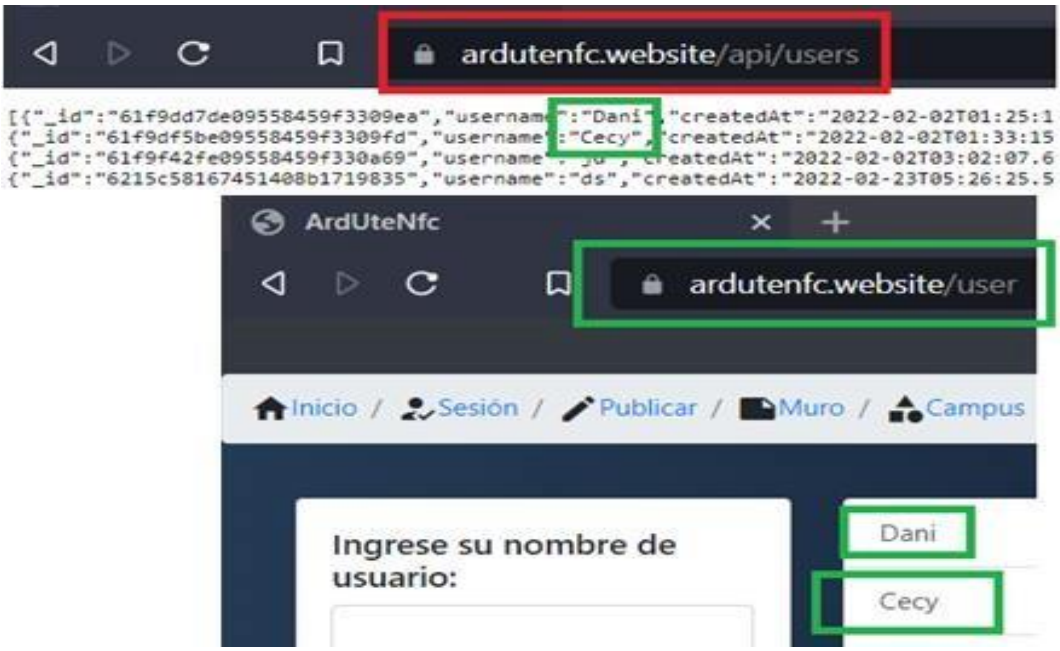

Figura 34. Ejemplo de ruta del servidor en formato *JSON*.

Rutas

Las rutas se usan para controlar la *REST-API*, que funcionan junto con la base de datos. En la Figura 35 se detalla las direcciones del sitio Web, en donde se pueden hacer peticiones, como obtener archivos estáticos.

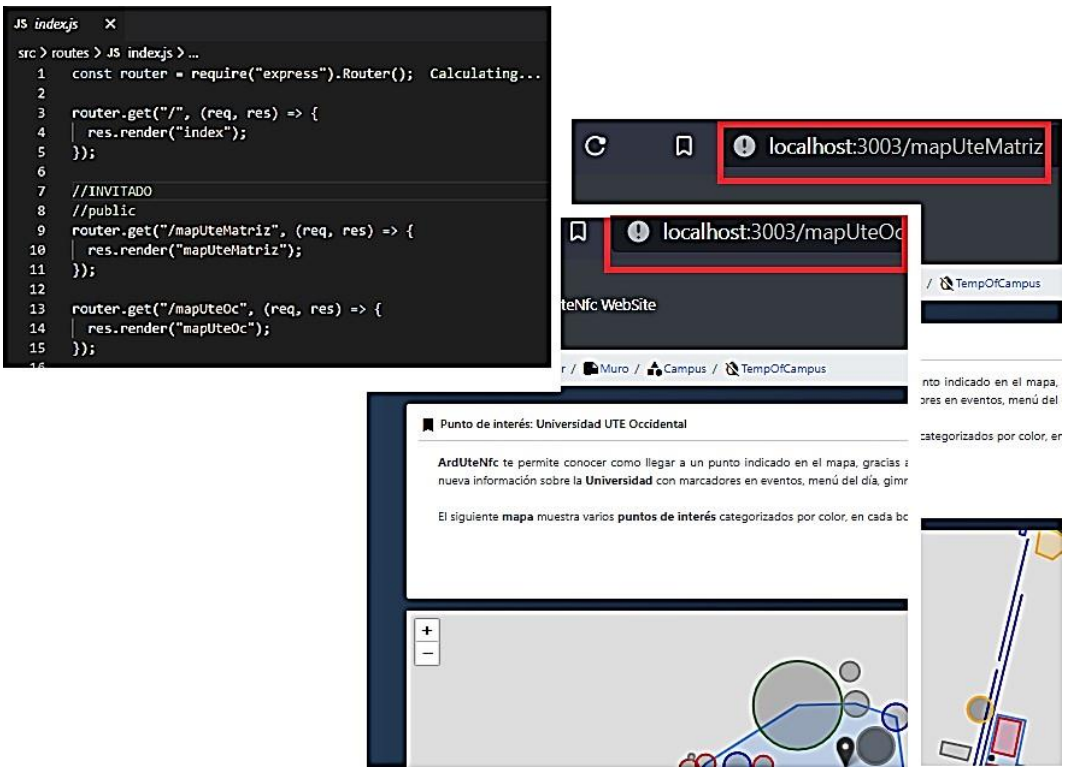

Figura 35. Ejemplo de rutas del servidor de la aplicación.

Rest-API

El servidor almacena una *REST-API* y esta definición de datos se encarga de enviar archivos al *Frontend* del navegador, por ejemplo, enviar imágenes, textos.

Las rutas se complementan con un sitio web (CRUD) y se manipulan para la presentación al usuario con el *framework React*. *js*.

En la Figura 36 se resumen cuatro funciones:

- Create \ Crear,
- Read \ leer,
- *Update* \ *actualizar* y
- Delete \ borrar del *CRUD*.

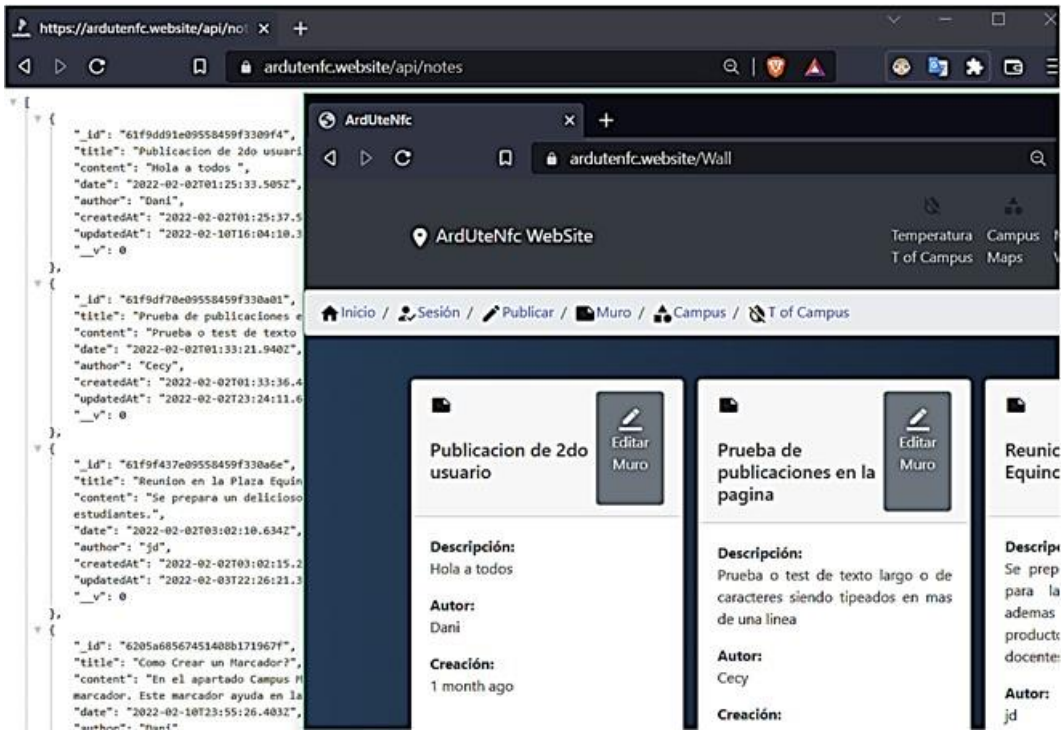

Figura 36. Ejemplo de REST-API usado en *ArdUteNfc Website*.

Archivos Estáticos

Los archivos estáticos se preparan en un sitio web para ser vistos por los usuarios. *Express* dispone de un módulo *static*, el cual define el archivo a encontrar.

La Figura 37 por ejemplo define una ruta. El sitio web presentainconvenientes, el primero es que los directorios no encuentran el archivo, pero se soluciona este problema con la función *Path*, presente en *Express*. Esta función une directorios con el servidor e ignora la unión de directorios de diferentes sistemas operativos como *Windows* o *Linux*.

```
25  //static files
26  app.use(express.static(path.join(__dirname, "assets/")));
27  app.use(express.static(path.join(__dirname, "public/")));
```

Figura 37. Ejemplo de uso de la función *Path* para unir directorios.

Framework *Mongoose*

Mongoose es necesario para modelar los datos, para lo cual se debe acceder usando el *framework Mongoose*. El modelo de almacenamiento con las variables de un contacto define el tipo de dato que pueden ser: texto, numérico, booleano, entre otros.

El esquema definido se observa en la Figura 38, que puede ser reutilizado las veces que se quiera, esto gracias al servidor creado usando *Node. js*. Finalmente, este módulo de conexión configurado se exporta y es utilizado por el archivo de configuración de rutas del servidor.

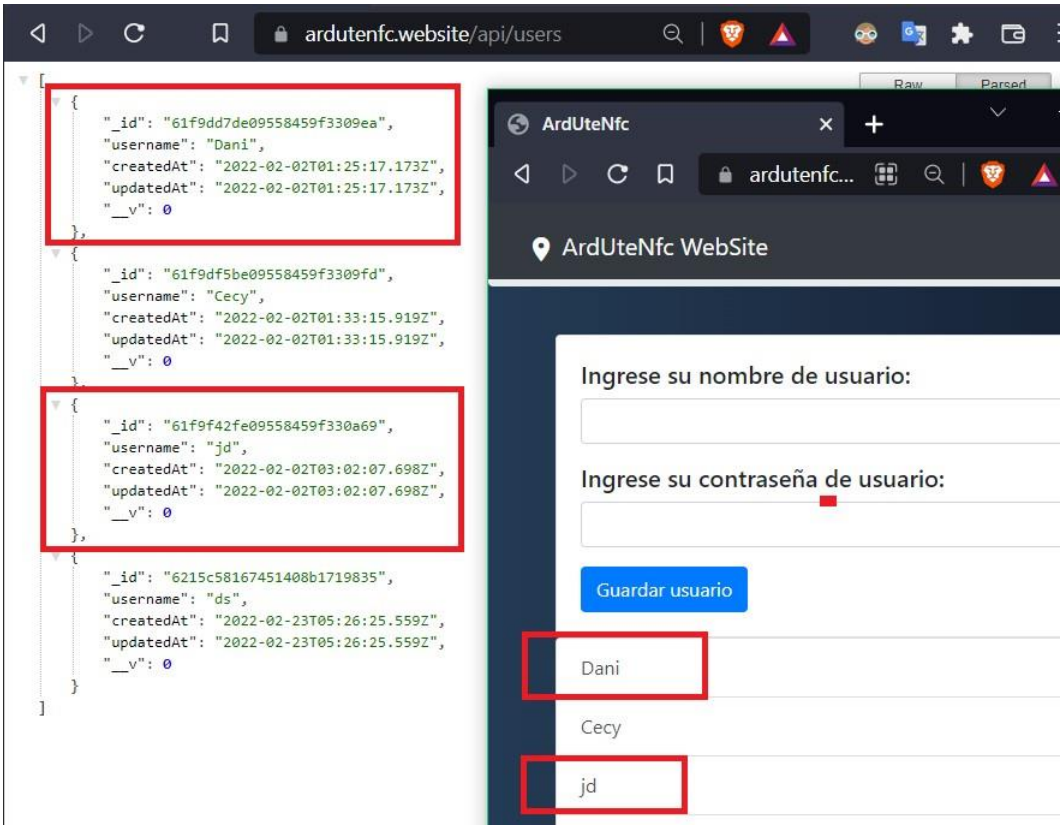

Figura 38. Conexión base de datos mediante *framework Mongoose*.

Implementación

Node. js

Para crear el servidor se usó *Node. Js* y se añaden más funcionalidades. Se inicia creando un archivo de configuración que permite iniciar e ir controlando las configuraciones básicas del servidor.

Iniciar un proyecto en *Node. js* se realiza mediante la terminal integrada de *Visual Studio Code*, para lo cual se creará una carpeta con el nombre del proyecto y se accederá mediante el comando "*npm init —yes*" en la consola.

Se crea un archivo denominado '*package.json'*, detallado en la Figura 39, en él se observan configuraciones durante el desarrollo y despliegue, como: nombre, descripción y scripts para la aplicación.

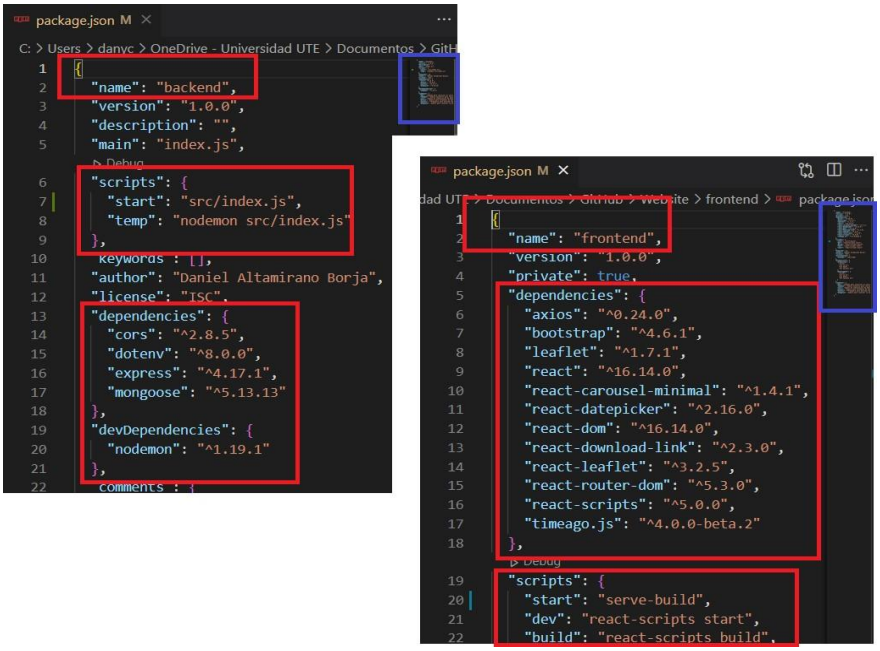

Figura 39. Ejemplo de '*package*.*json'* del proyecto *ArdUteNfc Maps Apps*.

Framework *Express*

El f*ramework Express* esencial en este *stack* nos ayuda a escribir una aplicación con un grado de calidad más alto, que permite tomar la decisión de ejecutarlo uno mismo. Es código testeado y probado que con seguridad funciona.

En la Figura 40 se observa una de las funcionalidades primarias. Se utiliza su biblioteca para escuchar y trasmitir por un puerto local.

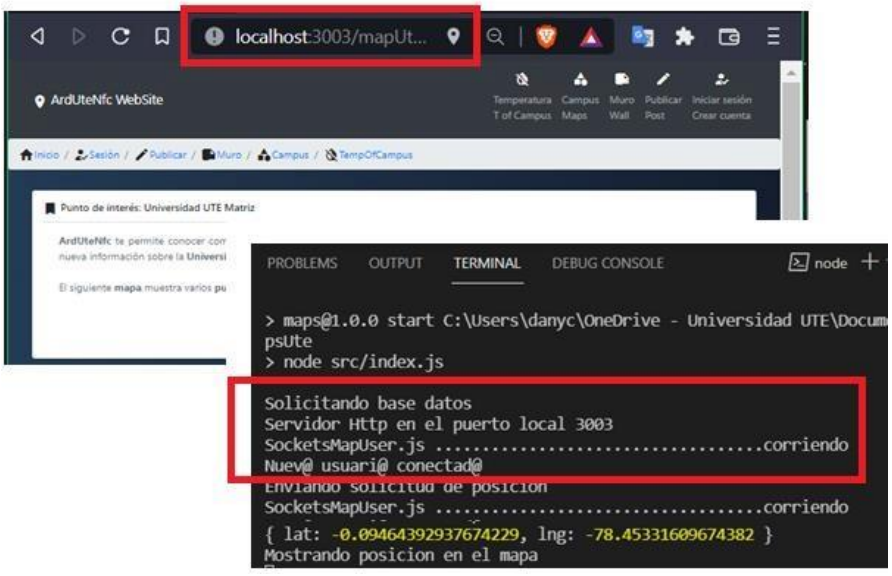

Figura 40. Ejemplo de servidor con *Express*.

Peticiones en las Rutas

Permiten en el servidor definir las rutas mediante el tipo de dato que solicita el usuario del sitio web. A continuación, se mencionan las peticiones usadas:

***Get*:** Obtener el dato de la base de datos mediante un identificador.

***Post*:** Crear el dato de la base de datos agregando un identificador único.

Se necesita de un identificador existente para los métodos en las rutas de la *REST-API*. Se visualiza en las líneas de la Figura 41.

***Put*:** Actualiza el dato de la base de datos mediante un identificador.

***Delete*:** Borra el dato de la base de datos mediante un identificador.

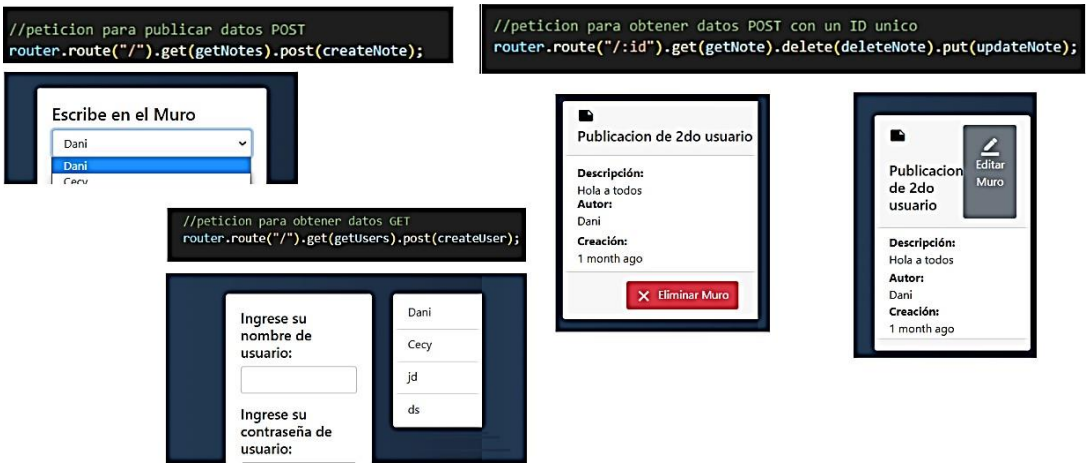

Figura 41. Ejemplo de rutas creadas para una nota mediante el uso de CRUD.

Localización con Leaflet .js

Se implementó el f*ramework Leaflet. js* en dos ubicaciones geográficas, la primera ubicación es la Universidad UTE, campus matriz; la segunda ubicación el campus occidental de esta Entidad Educativa.

Las ubicaciones servirán a docentes y estudiantes, tanto nuevos como antiguos, y a visitantes de la Universidad.

Este *framework* se utiliza para localizar. Cuenta con marcadores que implementan coordenadas geográficas, como latitud y longitud, que permiten ubicar ciertos objetos detectables como teléfonos y computadores. Además, cuenta con mapas e interfaces, por ejemplo, radares. Esta funcionalidad puede ser implementada para la comunicación bidireccional con el módulo *Socket.io*, que es eficiente usando *Websockets*. Los usuarios pueden hacer peticiones y solicitudes al servidor en tiempo real.

La construcción de objetos con *Leaflet. js* se observa en la Figura 42. El uso de parámetros de ubicación como latitud y longitud crean objetos como círculos y polígonos, que introducen coordenadas, al que se atribuye un estilo *CSS* al objeto antes de renderizarlo.

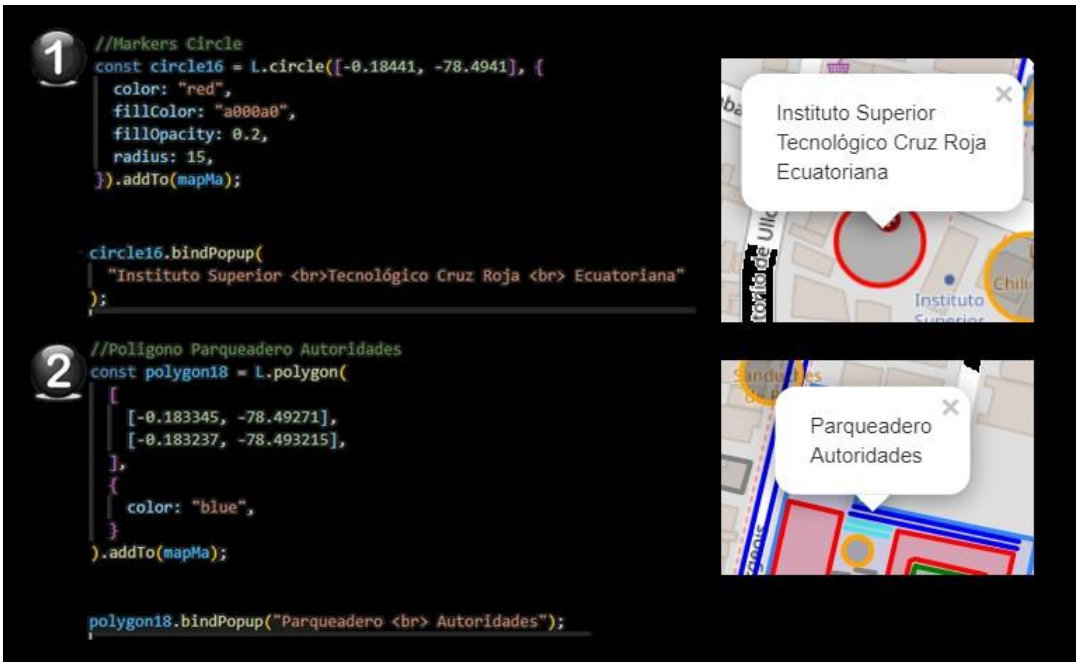

Figura 42. Ejemplo de construcción de recursos con *Leaflet. js.*

MongoDB NoSQL

Node. js permite mediante su biblioteca *Mongoose* conectarse a través del protocolo de *MongoDB* a una base de datos local y/o en la nube. Proporciona nombre, eventos y errores de conexión con los datos de la base.

La configuración de *MongoDB* se ubica en el servidor como el directorio *Models*.

Framework Sockets.io

Websockets se utiliza para crear aplicaciones como chats y para ubicar a las personas en mapas como *Leaflet. js.*

El programa se instala con el administrador de paquetes de *Node .js, Npm,* luego se implementa el archivo de configuración inicial del servidor. Para este proceso se usa la Librería principal *Socket.io.*

La Figura 43 indica la creación de constantes para usar el *framework SocketIO*. Finalmente se establece la comunicación con el archivo de configuraciones a compartir con el servidor.

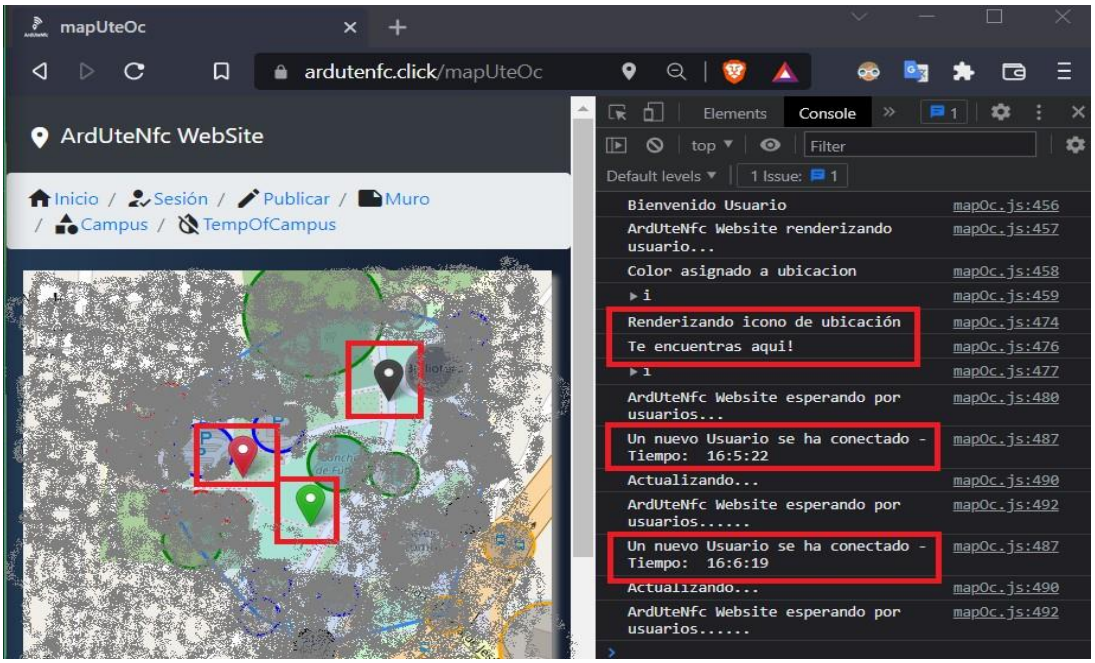

Figura 43. Ejemplo de configuración del servidor para archivo sockets.js.

Framework React .js

Utiliza un servidor conocido por el lenguaje JSX, bajo el cual funciona en un puerto el *frontend,* desde un navegador mediante la Figura 44 se pueden incorporar varios módulos necesarios parar reutilizarlos en forma de componentes, y renderizar la capa del frontend con el backend relacionada con los datos de usuario final.

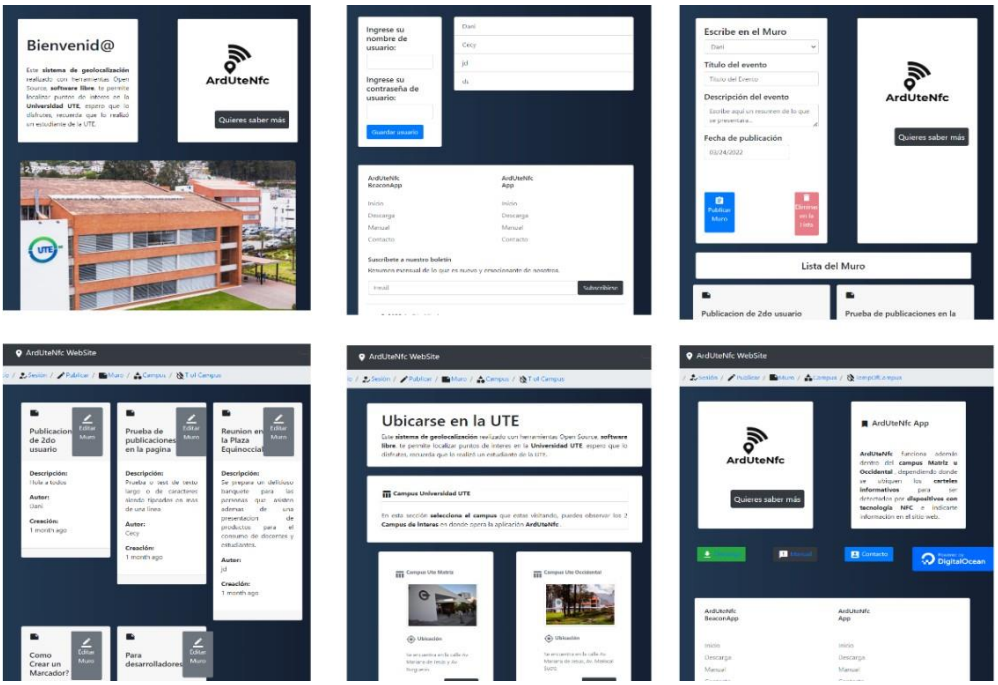

Figura 44. Componentes reutilizados en el proyecto.

Framework Axios

Axios es un Middleware usado para la conexión de recursos disponibles en otros dominios, puertos locales bajo el protocolo *HTTP*, *HTTPS*, entre otros, que permiten usar promesas para la obtención y envió de datos. Como se detalla en la Figura 45 se usa para conectar la aplicación *Backend* que usa *MongoDB* con la aplicación *frontend* que utiliza *React. js* para renderizar contenidos.

```
getNotes = async () => {
  const res = await axios.get("https://ardutenfc.website/api/notes/");
  this.setState({
    notes: res.data,
  });
  console.log("Lista de Publicaciones");
  console.log(this.state.notes);
};

deleteNote = async (noteId) => {
  await axios.delete("https://ardutenfc.website/api/notes/" + noteId);
  this.getNotes();
  console.log("Publicacion eliminada.");
```

Figura 45. Ejemplos de *Axios* solicitando recursos del *backend*.

Framework Ngrok

Para la salida a Internet y pruebas de este proyecto, al inicio, se usó *Ngrok*, es un *Software* que permite transformar la dirección pública en una dirección privada. Como se observa en la Figura 46, cuando se implementó *Websockets* con *Leaflet*. *js*, la información del usuario que visitaba la página web no era visible, por lo que tras verificar y encontrar el error se descubrió que usaba *HTTP* en lugar de *HTTPS*, esto llevó a un problema conseguir un certificado y una llave.

```
ngrok by @inconshreveable                                       (Ctrl+C to quit)

Session Status                online
Account                       danychet.dz@gmail.com (Plan: Free)
Version                       2.3.40
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://e3b4-201-183-101-33.ngrok.io -> http://localhost:
Forwarding                    https://e3b4-201-183-101-33.ngrok.io -> http://localhost

Connections                   ttl     opn     rt1     rt5     p50     p90
                              12      0       0.06    0.03    5.32    10.28

HTTP Requests
-------------

GET /static/media/occi0.59fdbf374f2dc1fbe347.jpg                    200 OK
GET /static/media/occi2.f175365173f5a56115bf.jpg                    200 OK
GET /static/media/occi3.0ce094e58decc149213e.jpg                    200 OK
GET /static/media/matriz2.32541f136ea0ee3d21f1.jpg                  200 OK
GET /static/media/matriz1.e2fa1850aad4d9e6bf7a.jpg                  200 OK
GET /static/media/occi.cffe5f01864525c2398e.jpg                     200 OK
GET /static/media/matriz3.90e855bd9678d2aaad07.jpg                  200 OK
GET /src/img/locationWh.png                                         200 OK
GET /static/media/BibliotecaUteOccidental.557cd84c9e392b9dbce7.jpg  200 OK
GET /static/media/occi0.59fdbf374f2dc1fbe347.jpg                    200 OK
```

Figura 46. *Ngrok* testeando rutas inseguras mediante *HTTP*, *HTTPS*.

3.1.3. Tablero 3: Fase ArdUteNfc Apps

El tablero para el desarrollo de Apps en *Android Studio IDE* resume dos aplicaciones móviles planteadas:

- *ArdUteNfc Tag App* y
- *ArdUteNfc Beacon App.*

El entorno de desarrollo de *Android Studio* se escogió una interfaz común de usuario. Se usó la versión 28 y 30 para la compatibilidad.

Análisis de requisitos

El desarrollo separado de aplicaciones móviles fue necesario para proceder con la fase de localización, caso de *beacon*s y etiquetas NFC, esquema de arquitectura la Tabla 13.

La lectura y/o escritura en *Tags NFC* se presentará mediante una cadena configurada en diferentes pantallas, una para la lectura y otra para la escritura.

Tabla 13. Arquitectura de aplicaciones ArdUteNfc

Desarrollo ArdUteNfc Apps	Tag App	Beacon App
Menús	X	X
Lectura	X	X
Escritura	X	X
Configuraciones		X

En la Figura 47 se visualiza el desarrollo de ambas aplicaciones. El producto final se lo puede revisar en la sección de aplicaciones del sitio web *ArdUteNfc*.

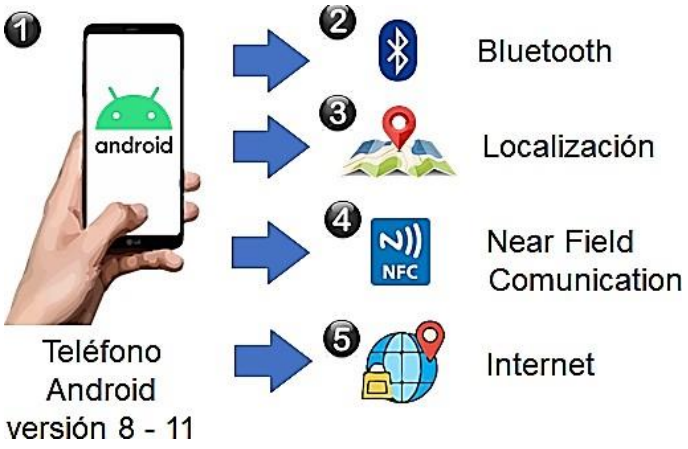

Figura 47. Recursos necesarios en el manifest para aplicaciones de *ArdUteNfc*.

Como se puede observar en la Figura 48, el tamaño de un beacon es igual al de una moneda de diez centavos, y el chip se basa en tecnología *bluetooth*, permite información por este protocolo.

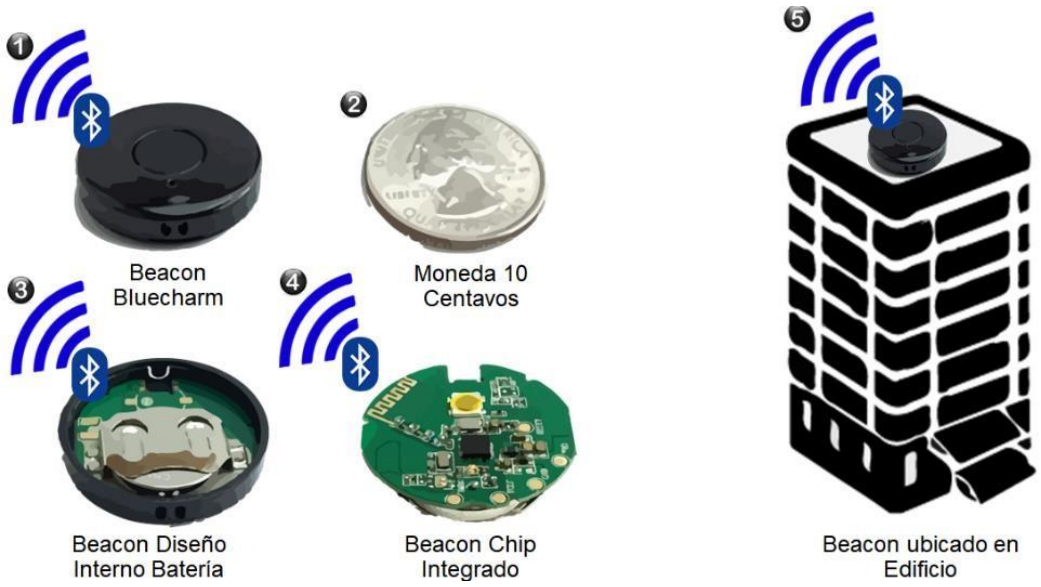

Figura 48. Características del *Beacon Blue Charm* en un entorno vertical.

Instalación del *Beacon Blue Charm*

El primer paso es descargar la aplicación *Android*, *Eddystone* desde la *PlayStore* y ejecutarla. Segundo paso, presiono el dispositivo *beacon*, durante dos segundos hasta que parpadee la luz Led de color verde, luego se debe soltar el botón.

Una vez encendida la luz se puede realizar una conexión para configurar parámetros en el *beacon* durante dos minutos mediante la aplicación móvil *Eddystone*. Si se agota el tiempo se debe reiniciar el b*eacon*, presionándolo para apagarlo y luego para prenderlo nuevamente.

Configuración de la aplicación *Eddystone*

El s*oftware* oficial de la marca *Eddystone* permite proteger el *beacon* adquirido con una nueva contraseña, además del nombre del *beacon* a emitir los parámetros bluetooth como el alcance de la señal mediante *RSSI*. Una vez que se guarden las configuraciones el *beacon* se apagará, este funcionamiento es necesario para dejar de emitir una señal.

El *beacon* tras dos minutos de espera de configuración, entra en modo de funcionamiento programado por el usuario. La lectura de b*eacons* se presenta mediante una cadena de lectura y/o escritura configurada en diferentes pantallas, una para la búsqueda y otra para su administración.

En administración se puede realizar acciones cuando se frecuenta un *beacon* en un lugar específico, para ello se incluye un radar amigable para el usuario.

Diseño lógico

La primera aplicación en *Android*, *ArdUteNfc App* cumplió la función de lectura y escritura de *Tags NFC*. Permitió la conexión de *Hardware RFID*, que estaba presente en las tarjetas o *Tags* usados en *ArdUteNfc IoT*.

Mediante la Figura 49 se demuestra las etapas del protocolo *NFC* disponible en ciertos dispositivos *Android* y la construcción del esquema de la aplicación para la comunidad universitaria.

Figura 49. Etapas de escritura de un *tag* mediante la app.

La finalidad de la segunda aplicación se resume en la Figura 50 permitiendo buscar *beacons Blue Charm,* y la información detallada acerca de estos dispositivos encontrados.

Figura 50. Elementos propuestos en el ambiente de uso de la app.

Se necesitó el protocolo bluetooth, mediante Android Studio se puede acceder a estas herramientas para el desarrollo entre otras como el acceso a la ubicación, vibración del dispositivo y otros.

Diseño físico

Permisos *NFC*

Se dispone de un tag. bajo el estándar *ISO 1443* y un teléfono con tecnología *NFC*. Al usar *Android IDE* es necesario conocer la información de la etiqueta como su *UUID*, este valor se encuentra escrito en valores enteros y numéricos. La Figura 51 señala la línea de código necesaria para el permiso del protocolo *NFC* y derivados.

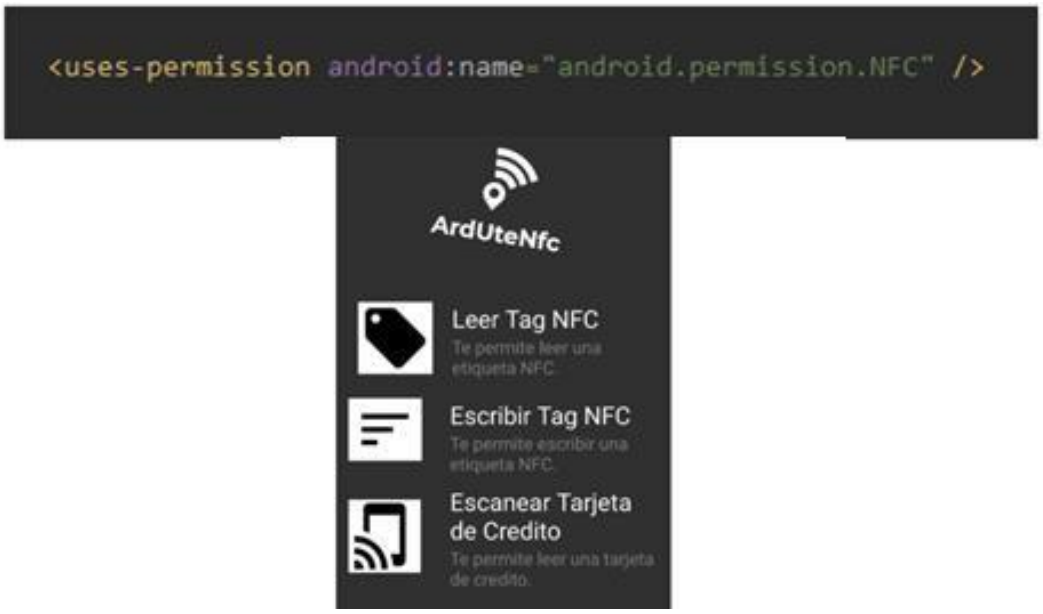

Figura 51. Sentencia AndroidManifest para usar el protocolo *NFC*.

Permisos Bluetooth

Se dispone de cuatro b*eacons* de la marca *Blue Charm*, por otra parte, se necesita el teléfono con bluetooth. Como se indica en la Figura 52 al usar *Android IDE* permite conocer la información bluetooth del dispositivo para la ubicación. En el manifest se coloca el permiso para bluetooth y ubicación.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Figura 52. Sentencia *AndroidManifest* para usar el bluetooth del dispositivo.

Permisos Ubicación

Se dispone del dispositivo *Android* y acceso a la ubicación. En la Figura 53 se observa que cuando se usa *Android IDE* es necesario conocer la información del dispositivo para la ubicación en el manifest que coloca el permiso para utilizarlo en la App.

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

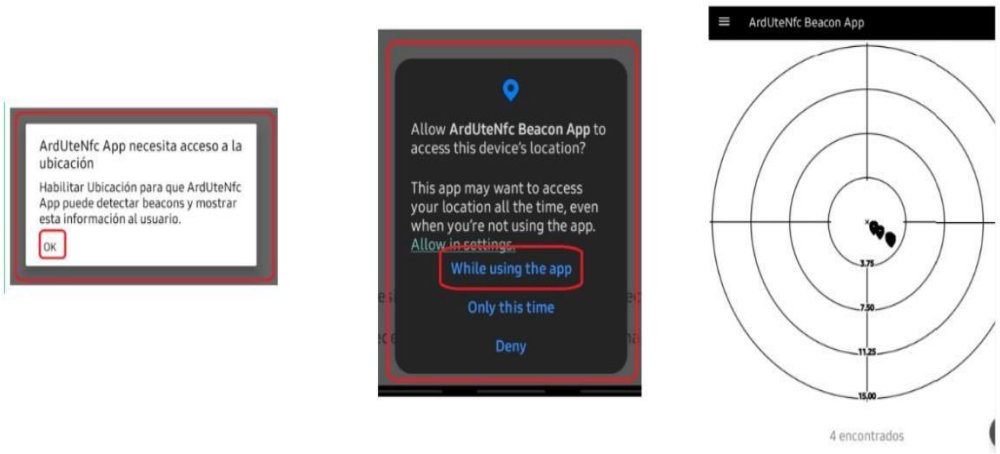

Figura 53. Sentencia *AndroidManifest* para usar la ubicación del dispositivo.

Implementación

ArdUteNfc Tag App

La aplicación *ArdUteNfc Tag App* necesitó la lectura y/o escritura sencilla, mediante un dispositivo *Android* con *NFC,* que gracias a la proximidad del campo magnético del Tag *NFC* permite leer y escribir en él.

Lectura de *tag NFC*

La Figura 54 visualiza *nfcRead* y en su archivo *XML* se inserta el diseño con un elemento *ImageView* y un elemento *TextView* con un Id implementado como *tvNfcMessage*. Renderizando la apariencia del texto corto o largo consigue compatibilidad con varios dispositivos *Android*.

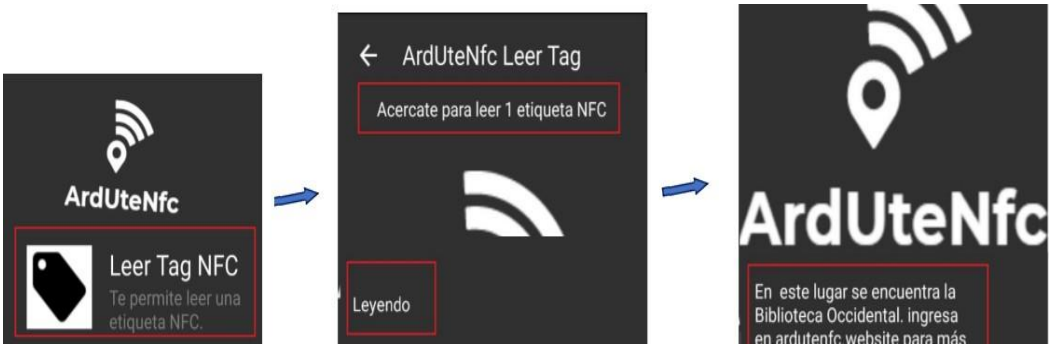

Figura 54. Desarrollo del *frontend* Leer en *ArdUteNfc tag App*.

El resultado de aplicar la lectura de *tags NFC* es un cuadro de texto que se renderiza en el elemento *TextView*.

En la Figura 55 con ayuda de Android *Studio IDE* mediante el arreglo *NdefMessage* y el objeto Text con un Id *tvNfcMessage* se obtiene la información.

```
NdefRecord record = ndefMessages[0].getRecords()[0];

byte[] payload = record.getPayload();
String text = new String(payload);
tvNFCMessage.setText(text);
```

Figura 55. Sentencia usada para leer el *tag NFC* en *Android IDE*.

Escritura de *tag NFC*

La escritura de *tags NFC* inicia con la lectura del mismo archivo en el *AndoridManifest* sobre permisos *NFC*, pero en este caso, para una diferente

actividad se debe implementar Intent-Filter para iniciar *NFC* desde otra actividad compartiendo el recurso.

En la Figura 56 se pueden observar las líneas necesarias para implementar el descubrimiento de las tarjetas o *tags NFC*.

```
<intent-filter>
    <action android:name="android.nfc.action.TECH_DISCOVERED" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Figura 56. Sentencia usada para usar *NFC* en otra actividad de la *App*.

La lectura de *NFC* desde esta actividad se denominó *NfcWrite*. Los detalles de la construcción se visualizan en la Figura 57 y en su archivo *XML* se inserta el diseño que consta de un elemento *ImageView* y un elemento *TextView* que usa *Id evTagMessage*.

Figura 57. Desarrollo del *frontend* "*Escribir*" en *ArdUteNfc Tag App*.

La escritura en el tag se resume en las líneas de código presentes en la Figura 58. Fue posible implementar un control de errores en cada nuevo intento de escritura, es decir, cada vez que el intento de grabación en el *tag* era o no correcto.

Existen resultados del *frontend* sobre el funcionamiento explicado sobre lectura y escritura, hasta el momento. Consta más contenido ilustratorio en los manuales disponibles en los "Anexos" y en el sitio web *'ardutenfc.website'*.

```
if (messageToWrite != null && (!TextUtils.equals(messageToWrite,  b: "null")) &&
        (!TextUtils.isEmpty(messageToWrite))) {
    NdefRecord record = NdefRecord.createMime(messageToWrite, messageToWrite.getBytes());
    NdefMessage message = new NdefMessage(new NdefRecord[]{record});

    if (writeTag(tag, message)) {
        Toast.makeText( context: this, ("Successfully written to Tag !"), Toast.LENGTH_SHORT).show();
        finish();
    } else {
        Toast.makeText( context: this, ("Error Writing Message !"), Toast.LENGTH_SHORT).show();
    }
} else {
    evTagMessage.setError("Please enter the text to write");
    /*Toast.makeText(this,"Please enter the text to write",Toast.LENGTH_LONG).show();*/
}
```

Figura 58. Sentencia usada para escribir el tag *NFC* en *Android IDE*.

ArdUteNfc Beacon App

La aplicación *ArdUteNfc Beacon* necesitó permisos sobre la ubicación y bluetooth del dispositivo *Android*. A continuación, se demuestra la configuración lograda para el acceso a la ubicación y bluetooth.

Localización de un *Beacon* con *Android*

El funcionamiento de un *Beacon* es captado por cualquier tipo de celular con Bluetooth y la aplicación desarrollada permite detallar esta información, basándose en el protocolo Bluetooth activado y la ubicación en tiempo real del dispositivo.

En la Figura 59 se revela que el camino de la información del *Beacon* es estático, por lo que el usuario deberá escanearlo.

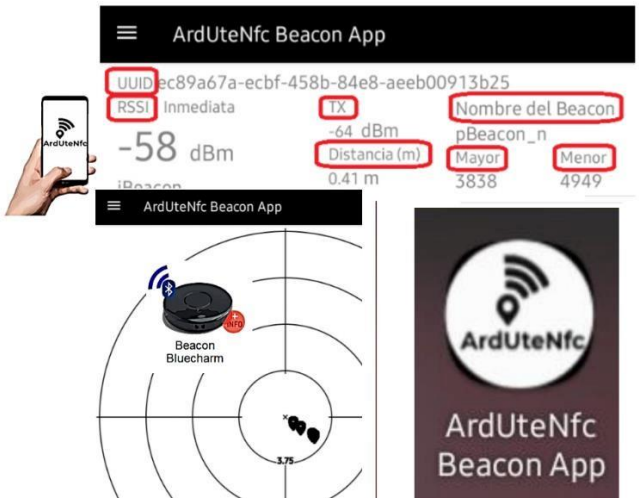

Figura 59. Rangos de *beacon Blue Charm* y dispositivo *Android*.

Lectura de *Beacon*

La lectura de *beacons* inicia con la actividad *DetectedBeacon,* resumida su elaboración en la Figura 60. El archivo *XML* inserta el diseño con un elemento *ViewStub* con Id *emptyScanView*, renderizando la apariencia de la vista al implementar los descubrimientos alrededor.

Figura 60. Desarrollo del *frontend* "Leer "en *ArdUteNfc Beacon App*.

El resultado de aplicar la lectura *de un beacon* mediante bluetooth, visualmente es un cuadro de texto que renderiza el elemento *ViewStub* que viene a ser el objeto detectado como en la Figura 61.

En lugar de observar la pantalla vacía por defecto, ésta cambiará con ayuda de programación de la variable *ViewStub* en pantalla. Con *Android IDE* se logró mediante el objeto *DetectedBeacon* visualizar los datos encontrados sobre b*eacons*.

```
public DetectedBeacon createFromParcel(Parcel in) {
    Beacon b = Beacon.CREATOR.createFromParcel(in);
    DetectedBeacon dbeacon = new DetectedBeacon(b);
    dbeacon.mLastSeen = in.readLong();
    return dbeacon;
}
```

Figura 61. Sentencia usada para leer el *beacon* en *Android IDE*.

Existen resultados del *frontend* sobre el funcionamiento explicado para lectura de b*eacons,* hasta el momento. También hay más contenido ilustrado en manuales disponibles en anexos y en el sitio web *'ardutenfc.website'*.

Administración de beacons encontrados

La administración de *beacons* inicia con la actividad *IManagedBeacon* resumida en la Figura 62 y en su archivo *XML* se inserta el diseño con el elemento *ViewStub* con Id *emptyScanView*.

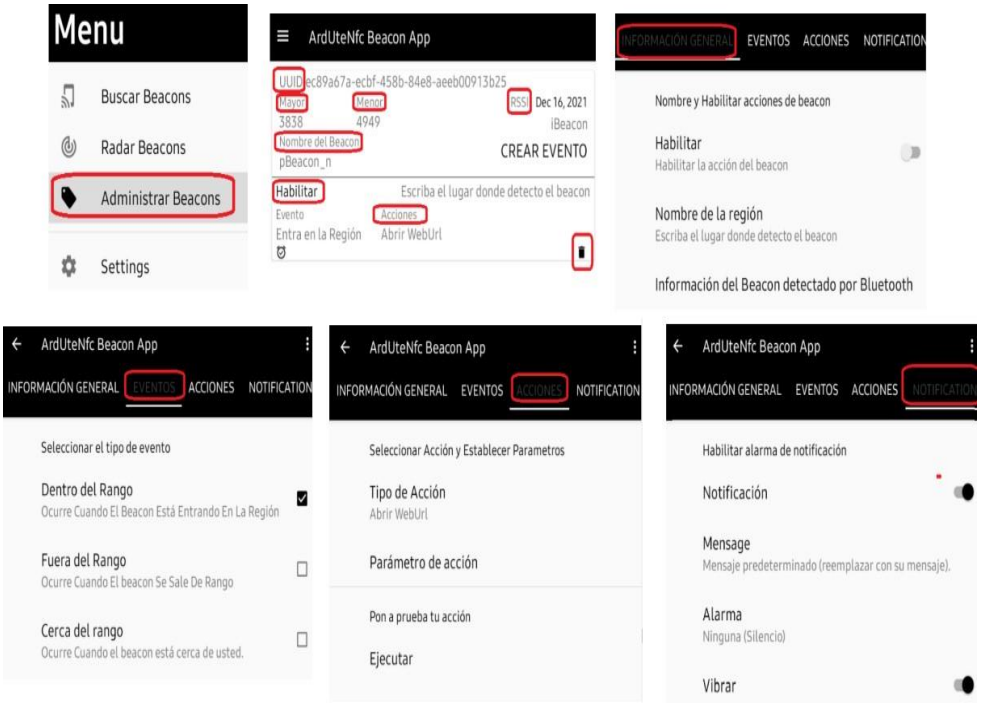

Figura 62. Desarrollo del *Frontend* “Administrar” en *ArdUteNfc Beacon App*.

El resultado de aplicar la administración de los *beacons* es usar varios de los objetos que identifican a uno en la lectura y emisión de sus parámetros, como se indica en la Figura 63; mientras exista conectividad se puede programar eventos como los realizados en esta aplicación en *Android*.

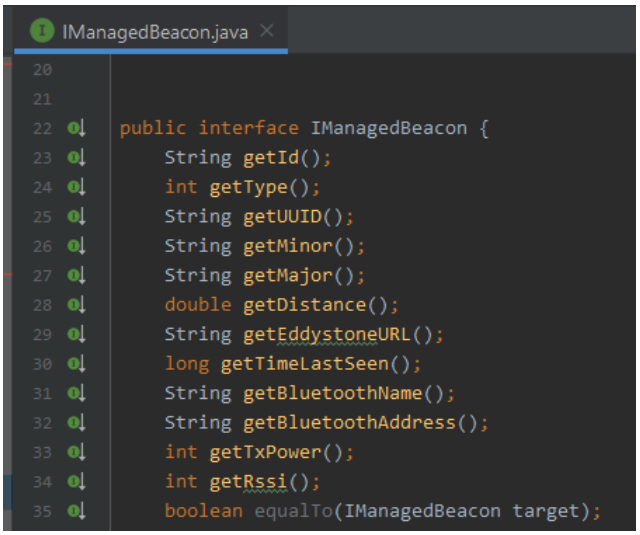

Figura 63. Objetos usados en la actividad *IManagedBeacon* de la app *ArdUteNfc Tag*.

3.1.4. Tablero 4: Fase ArdUteNfc *Despliegue*

La aplicación del tablero Despliegue permitió unir cada tablero construido previamente y presentarlo al uso de la comunidad en los diferentes campus.

Análisis de requisitos

ArdUteNfc IoT

El sistema *ArdUteNfc IoT* comenzó a funcionar después de usar varios programas y f*rameworks*. Al efecto *Arduino IDE* tiene un programa denominado "*Firmata"*, el cual se usó para implementar lecturas sobre:

- Sensor de temperatura,
- Tarjetas *NFC*.

Necesarios para aplicar este programa antes de correr el puerto serial desde *Node-RED.*

La implementación de las lecturas del sensor en *IBM Cloud* fue gratis durante treinta días, por lo que se procedió a actualizar para las pruebas el registro de la organización como se observa en la Figura 64, y en el registro de la nueva organización se utilizó un Id con servicios gratuitos.

En el momento de la activación del servicio *IBM Watson IoT* se aplicó el período de prueba de treinta días y se evidenció funcionalidades como el uso de tableros para la metadata del microcontrolador convirtiéndolo en *IoT*.

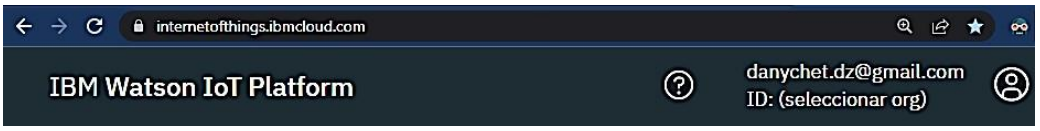

Figura 64. Servicio *IBM Watson IoT Platform* sin registro.

IBM CLOUD

Es la nube para el microcontrolador y sus señales. Es esencial una cuenta para acceder a funcionalidades como el renderizado de la información del microcontrolador en el sitio web *IBM Cloud*. Los pasos para acceder al servicio de temperatura en la nube se explican a continuación.

Crear una cuenta de *IBM Cloud*

Acceder a este tipo de cuenta se lograría mediante pago o de manera gratuita. Cabe explicar que al utilizar la versión de prueba de 30 días limitaría el número específico de dispositivos, así como para el desarrollo de este *Software* se usa la cuenta gratuita.

Crear un dispositivo en *IBM Cloud*

Se realiza mediante el módulo *Watson IoT.* Este módulo permite salir a Internet, específicamente a la nube *IBM*, para lo cual se debe detallar las características del microcontrolador: nombre, descripción, contraseña del dispositivo.

Estos datos son necesarios guardarlos con *Node–RED,* para su uso, más adelante.

Establecer la comunicación del dispositivo con *Node-RED*

Se realiza la compilación del archivo *Firmata* en el microcontrolador. En *Node-RED* se debe asignar un puerto serial y en el caso de *Windows* usar *COM3* o el que esté disponible.

De aquí en adelante se establecerán mediante flujos los pines y tipos de datos que emite el *ESP*-01 que son analógicos.

Conectar el *IBM Cloud* con *Node-RED*

Es necesario desde el administrador de módulos de *Node-RED* instalar *IBM Watson*. Este paso es fundamental para conectar el microcontrolador a la *IBM Cloud*.

ArdUteNfc Website

Pm2

PM*2* en el sitio web es necesario para administrar los procesos de las diferentes aplicaciones en los diferentes puertos. Una vez que se realicen las pruebas y se observe que los recursos de las páginas web funcionan bien sin *SSL.*

Cabe señalar que toda aplicación de Internet debe disponer *SSL*, requisito indispensable. Así que se procederá a comprar un dominio, se escogerá su nombre y se cancelarán sus valores correspondientes.

El dominio que se escoge en este proceso se maneja mediante la instalación de *Nginx* y *LetsEncrypt*. Para administrar y encriptar se realiza mediante configuraciones en los archivos del servidor.

El despliegue de *ArdUteNfc Website* se realizó en la plataforma *Digital Ocean* junto con sus *dropplets*. Se escogió un ordenador *Ubuntu* detallado en la Figura 73, explica su instalación. Las características esenciales de *Ubuntu* ejecutan *Node. js, PM2 y programas adicionales.*

Dominio SSL

Finalmente, las aplicaciones en línea necesitan un dominio que los identifique y sea accesible y recordado por los usuarios, a continuación, una lista de los dominios adquiridos en la Figura 65.

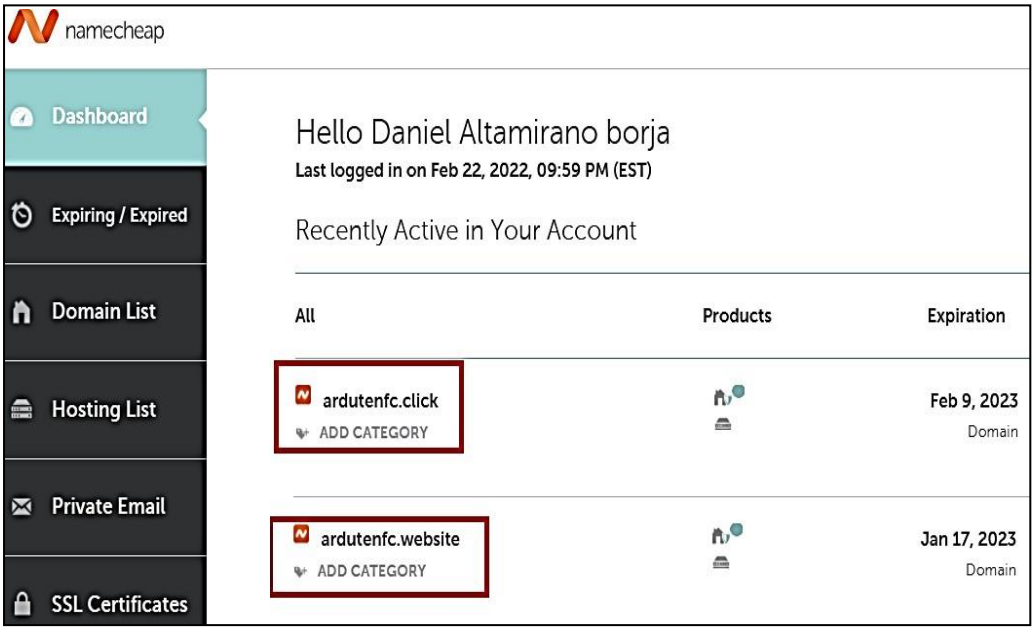

Figura 65. Dominios comprados para el proyecto *ArdUteNfc*.

Diseño lógico

ArdUteNfc IoT

En la Figura 66 se puede mencionar el camino de los datos hacia el computador host y finalmente a *IBM Cloud* usando como elementos los flujos y sus formatos de renderizado de información en la nube.

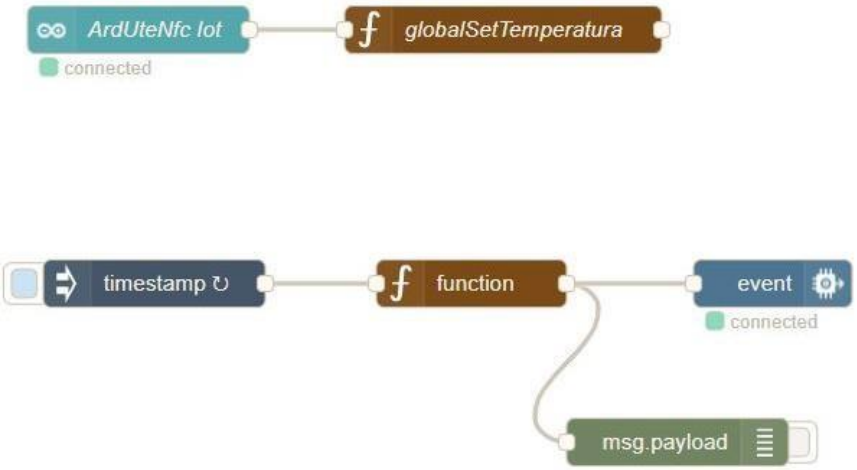

Figura 66. Flujos programados mediante *Node-RED* para la conexión en *IBM Cloud*.

El enlace proporcionado por *IBM Cloud* se puede interpretar en la Figura 70. Usa un link *HTTPS*, seguro por sus siglas para presentar los contenidos sobre temperatura en el campus de la Universidad.

Registro en *IBM Cloud*

Una vez que se conecta el microcontrolador al ordenador se reciben las señales, para lo cual es necesario registrar el dispositivo con la clave en la nube para visualizar los datos.

El registro permite acceder a funciones como el uso de paneles, datos guardados por el sensor, entre otras. En la Figura 67 se observa el módulo *Arduino Wi-Fi* conectado al ordenador, lo cual permite emitir señales analógicas de temperatura al servicio de *IBM Cloud*.

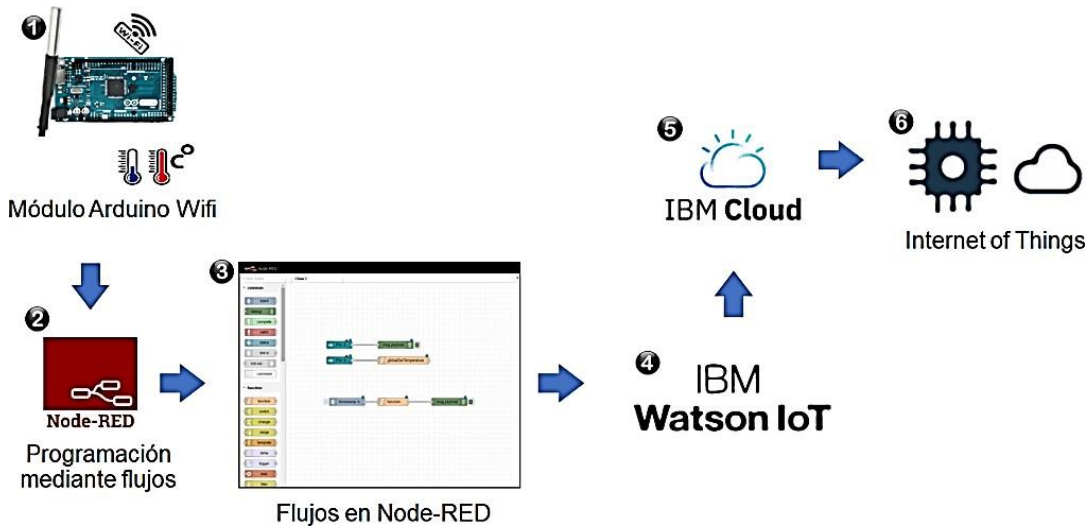

Figura 67. Módulo *Arduino Wi-Fi* trasmitiendo en tiempo real y online.

ArdUteNfc Website

El modelo del proyecto en el despliegue se puede visualizar en la Figura 68 que utilizando el framework *Express* nos permitió crear nuestro código de *backend* rápido. Se utilizo en lo posible la multiplataforma para continuar con la implementación del *MERN stack*.

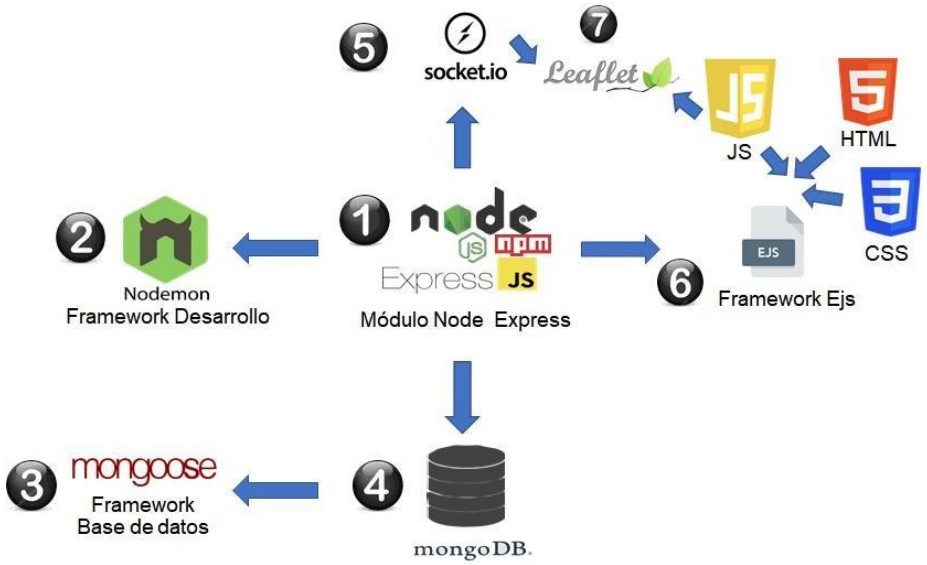

Figura 68. Modulo *Node .js, Express, Ejs* y *frameworks* adicionales.

Diseño físico

ArdUteNfc IoT

Un ejemplo sobre la presentación de datos obtenidos, se aprecia en la Figura 69, en el cual se puede estudiar y pronosticar las variables de la temperatura tanto mediante un histórico diario como mensual.

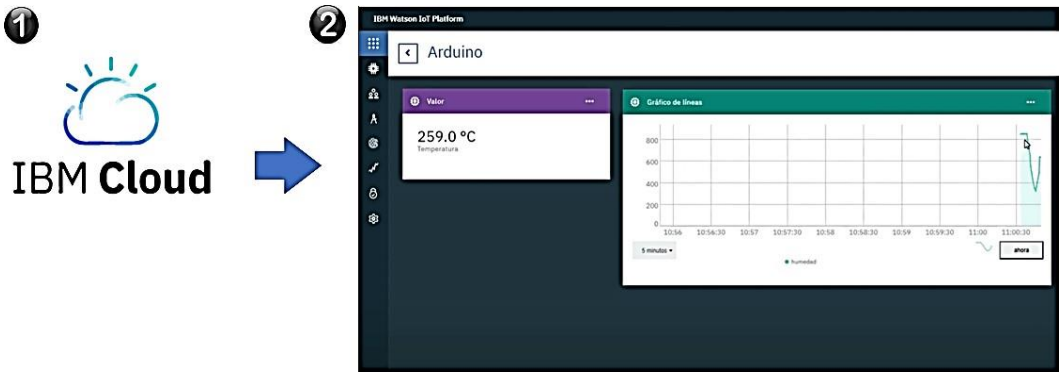

Figura 69. Paneles configurados en el entorno *IBM Cloud*.

El despliegue de *ArdUteNfc IoT* visible en la Figura 70 se realizó montando el servicio de temperatura elaborado mediante *Node-RED.*

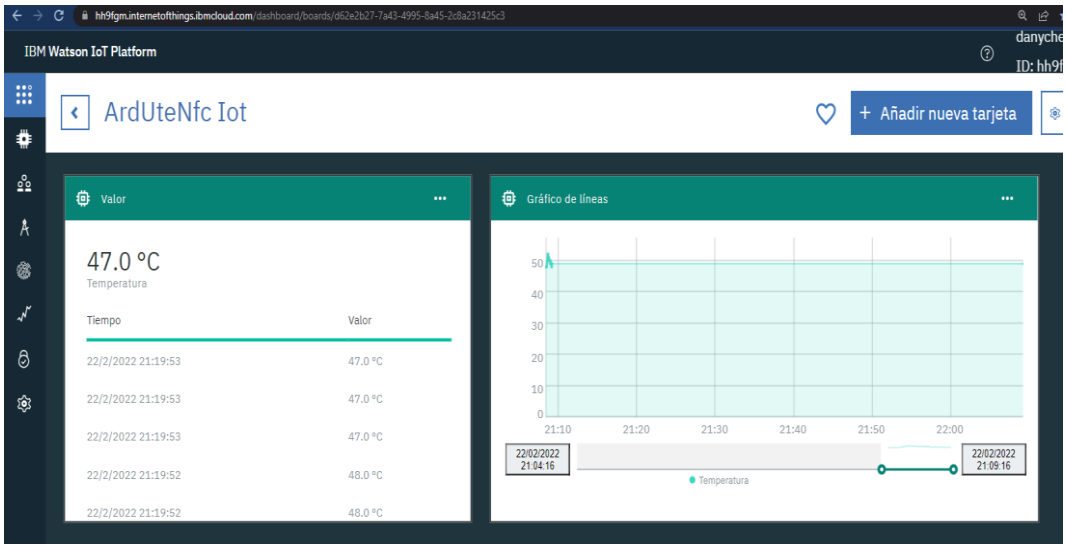

Figura 70. Despliegue de Información sobre la temperatura del sensor.

ArdUteNfc Website

Una vez instaladas las aplicaciones backend y frontend es necesario que el servidor de aplicaciones *PM2,* administre los diferentes puertos y ejecución en el D*ropplet*, como se visualiza en la Figura 71.

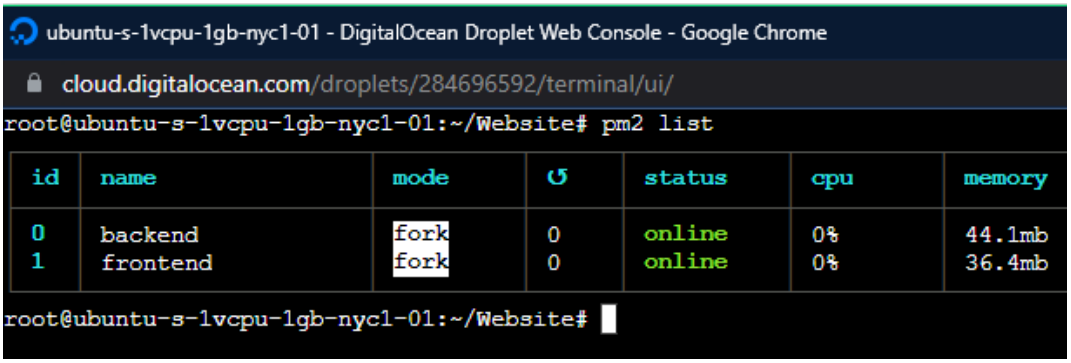

Figura 71. Listado de aplicaciones administradas por *PM2*.

Implementación

ArdUteNfc IoT

Con la implementación de *IBM Cloud* se puede realizar un seguimiento y monitoreo de los datos generados mediante interfaces que se implementan a partir de las lecturas del microcontrolador en la nube *IBM Watson IoT*.

El inicio del proceso *Node-RED* empieza en el puerto local con la adquisición de datos suministrados por la programación mediante flujos. El posterior uso en la nube se logra con la instalación y uso de la organización que administra el control de informacion del microcontrolador en la nube, ver Figura 72.

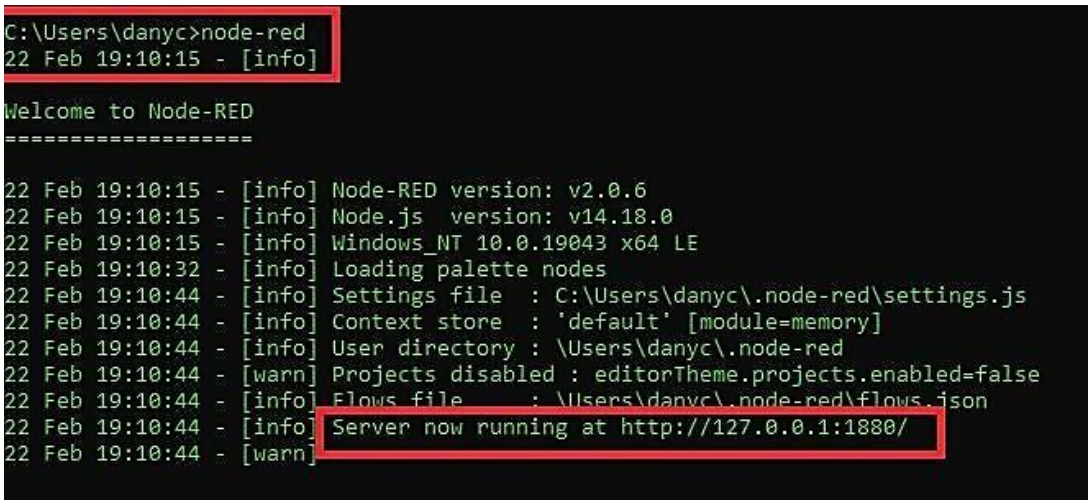

Figura 72. Ejecución de *Node-RED* desde la terminal.

Mientras el dispositivo se encuentre conectado a la red se puede hacer uso de la información generada en el medio ambiente por el microcontrolador.

ArdUteNfc Website

El despliegue del sitio web, se realizó mediante la adquisición de un servicio en la nube, se escogió Digital Ocean y se utilizo la característica de Dropplet, en la Figura 73 se observa la arquitectura de un sistema operativo Ubuntu.

Figura 73. Creación de dropplet para el proyecto Node. js.

Cuando el *Dropplet* arranca se debe configurar como si se tratara de un ordenador nuevo. En este caso se usa *Ubuntu 21.10* reflejado en la Figura 74 y todo se maneja mediante línea de comandos, facilitando la integración del *Software* usado.

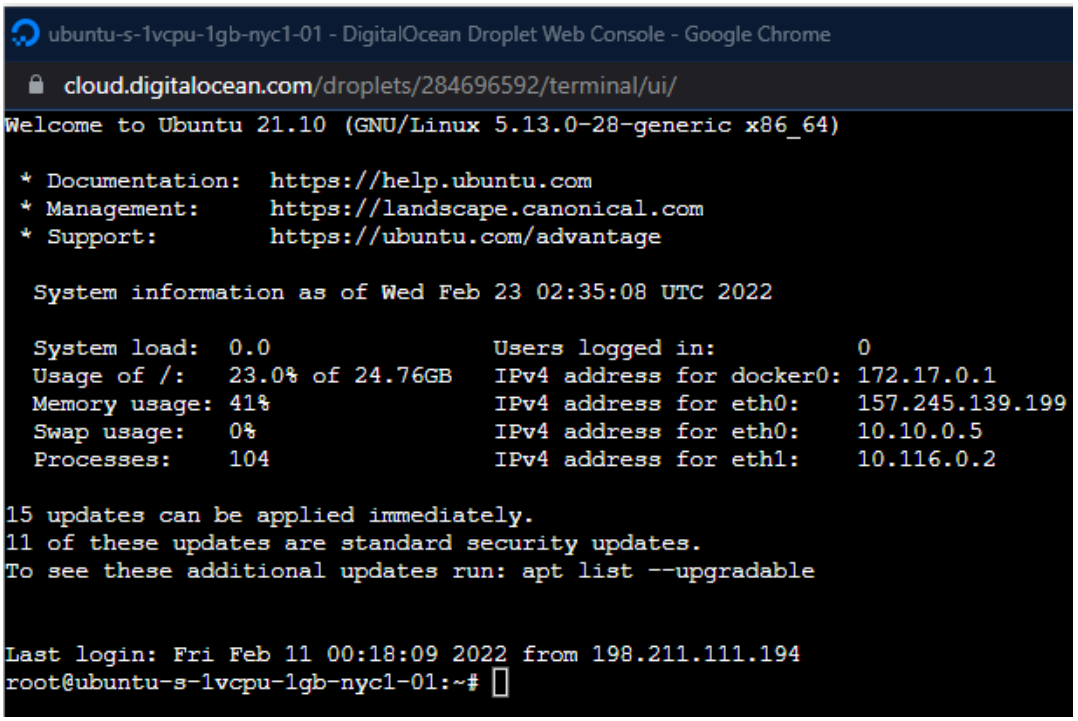

Figura 74. Dropplet iniciado localmente mediante sesión *SSH*.

Luego de realizar la conexión con el *dropplet* se puede instalar *GitHub* y clonar el proyecto del repositorio. Cuando la aplicación se copia con éxito se verifica el directorio como en la Figura 75.

Para acceder e ingresar el comando ‘npm install’, que lee el archivo Package.json e instala todas las dependencias del proyecto y lo ejecuta.

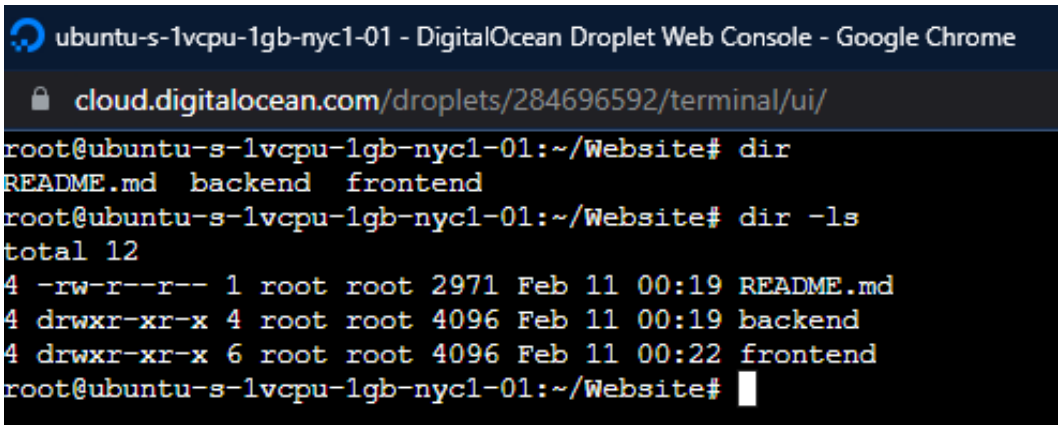

Figura 75. Directorio del proyecto listo en el *dropplet Ubuntu*.

Las aplicaciones se ejecutan correctamente y para eliminar el puerto asignado de la máquina que se controla en el *Dropplet*, es necesario usar *Nginx,* como en las líneas de código en el editor nano, de la Figura 76.

Administra las direcciones del sitio web desde diferentes puertos locales, usando Proxy para incluir el candado seguro con ayuda de *Namecheap y LetsEncrypt*.

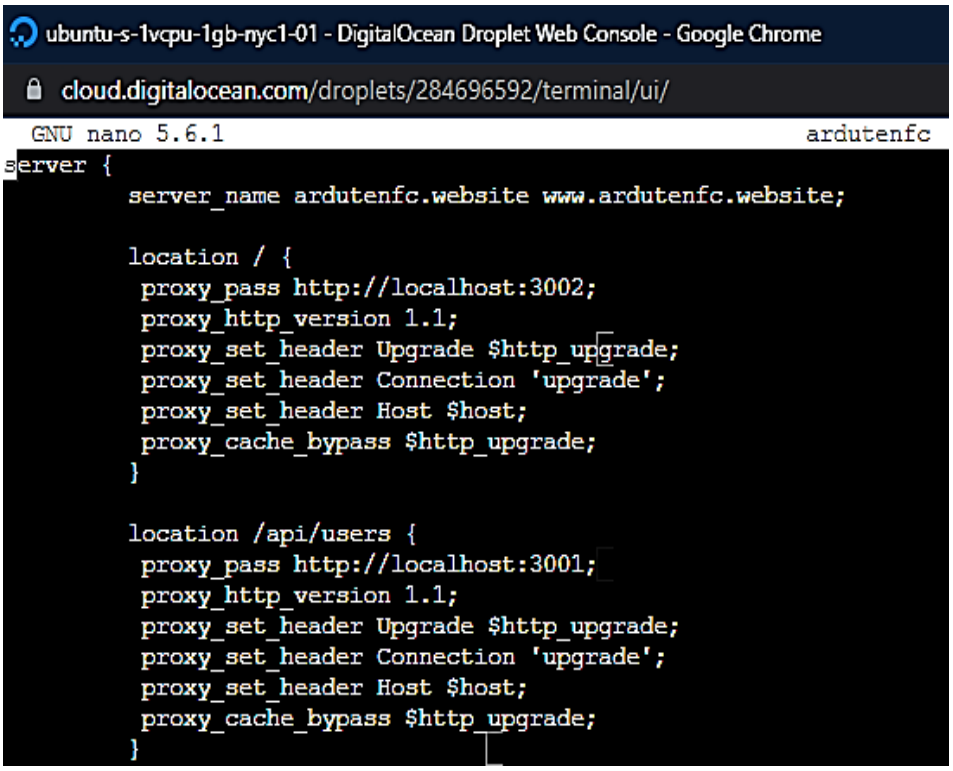

Figura 76. Archivo de configuración del servidor mediante Nginx.

El proceso de ingreso del dominio en el *Dropplet* se realiza desde el portal *Digital Ocean* y mediante el sistema operativo *Ubuntu*. El registro es sencillo y se indica en la Figura 77.

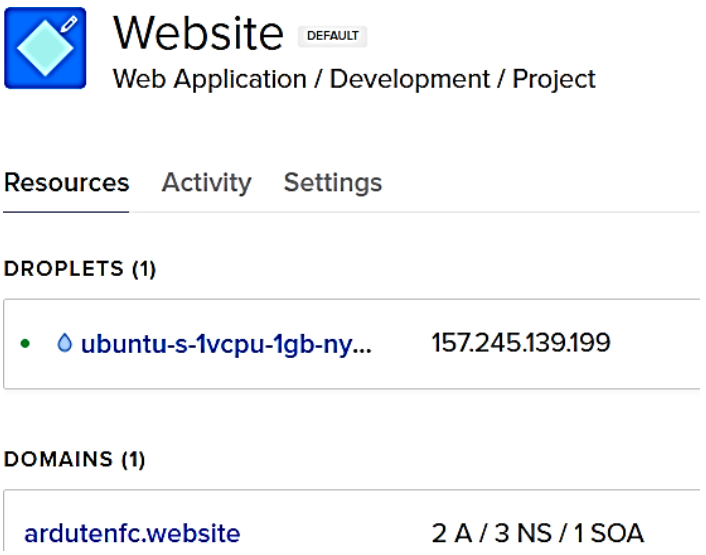

Figura 77. Ingreso del dominio comprado en *Dropplet* de *Digital Ocean*.

Luego el mismo archivo de configuración de *Nginx* ingresa la información del dominio y se realiza la implementación de *SSL* en el sitio web. Una vez concluido el proceso descrito en la Figura 78, mediante línea de comandos, estará listo con estándares de programación para recibir y enviar información con total seguridad.

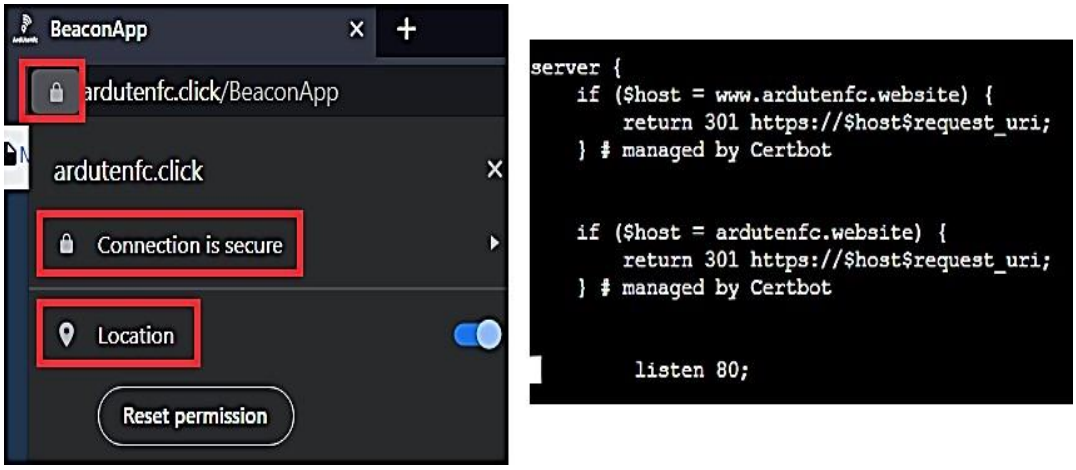

Figura 78. Código añadido al archivo del servidor *Nginx* para *SSL*.

ArdUteNfc Apps

El despliegue de *ArdUteNfc Apps* se realizó usando la herramienta propia de *Android* Studio como medio para generar archivos ejecutables o apks en el dispositivo *móvil*, esta decisión con el fin de simular las aplicaciones que serán parte del sistema de localización desde un hipervínculo dedicado a su descarga. Las aplicaciones serán parte de recursos estáticos del sitio web.

4. CONCLUSIONES Y RECOMENDACIONES

4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- La metodología *Kanban* ayudo *en* simplificar tareas extensas y repetitivas del proyecto mediante la reducción y construcción de un calendario mediante tableros, los cuales fueron de adquisión, pruebas, mejoras, despliegue entre otros y que mediante el trabajo adaptativo a lo largo de su resolución dieron un buen resultado.

- El sensor de temperatura, modelo *DS18B20* posee características como la durabilidad que resultaron suficientes para pruebas en cambios en la variable atmosférica. Se adapto perfectamente en presencia de diferentes sistemas, congelados como un refrigerador hasta el calor de una hornilla de cocina.

- Los dispositivos usados como *beacons* crearon localización vertical en edificios mediante el protocolo *bluetooth*, mientras que las etiquetas *NFC* se pueden clonar mediante una réplica de la tarjeta fácilmente, siendo un campo de estudio en futuras tesis sobre seguridad informática.

- El sistema elaborado es el resultado de la elección de un lenguaje actual, estándar y escalable, en este proyecto es *JavaScript*, la presentación de contenidos web y móviles sobre la Universidad UTE se logró mediante el uso de *frameworks como Express y React,* los cuales resolvieron todos los inconvenientes que presentó el sistema en cada una de sus etapas.

4.2 RECOMENDACIONES

- Se recomienda el uso de la metodología Kanban para proyectos pequeños y grandes, debido a la solución rápida y eficaz ante la incertidumbre con el uso de fases, recomiendo iniciar con el software en línea Kanbanchi para conocer las características que ofrece.

- Los dispositivos *beacons* se pueden reconocer mediante una aplicación cliente en este caso el sistema Android hace uso de una API para reconocerlos, además en el entorno físico se deben ubicar en lugares verticales para la identificación del objeto físico e inalámbrico creando un diseño de red para el protocolo bluetooth.

- El framework AXIOS ayudó a conectar los recursos *backend* y f*rontend* fácilmente, sin embargo, trabajé bajo los protocolos seguros e inseguros de la red en el desarrollo y despliegue del sistema en la plataforma *Digital Ocean* la cual ofrece muchas prestaciones para desarrolladores.

- El sistema desarrollado necesitará la implementación de nuevos servicios informáticos enfocados tanto para ordenadores y móviles. La comunidad universitaria quien se beneficiará necesitará el estudio de uso moderno con sensores y módulos mediante el avance del IoT.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

Aguilera Blanco, L. R. (2021). INTERNAL CONFIGURATIONS FOR THE HARDENING SECURITY IN NGINX.

Amaya Fariño, L. T. (2020). El IoT aplicado a la Domótica.

Briceño Portilla, W. A. (2021). mplementación de nueva tecnología front-end para mejorar el rendimiento de sitios web.

Buñay, P. E. (2020). Aplicación de la metodología kanban en el desarrollo del software para generación, validación y actualización de reactivos, integrado al sistema informático de control académico UNACH.

Chaparro, A. Y. (2021). Implementación de beacon para la visualización de las ofertas educativas de la Corporación Universitaria Unitec.

Chinga Olmedo, P. Y. (2021). Sistema IoT para el control de temperatura en zonas pecuarias.

Dávalos-Jiménez, A. P.-M.-F.-L. (2019). Simulador de negocios Capsim: planeación estratégica en la industria de los sensores.

Gaete, J. V.-R. (2021). Enfoque de aplicación ágil con Serum, Lean y Kanban.

Herrera Llori, W. A. (2017). Diseño y construcción de un prototipo para el control y monitoreo de registro de tiempos de transporte urbano mediante el uso de tecnología RFID, Arduino Mega y Raspberry Pi 2.

Ifrah, S. (2021). Deploy Docker Containers on GCP Compute Engine. In Getting Started with Containers in Google Cloud Platform.

Jiménez Calderón, M. (2020). Ingeniería de pruebas: aplicación, alcance y rendimiento en un banco de test para una aplicación cliente-servidor.

León Sillo, D. Y. (2020). Desarrollo de un sistema de Identificación por Radiofrecuencia para el control de inventarios de Activos Fijos en la Universidad Peruana Unión Filial Juliaca.

Llamas, L. (2017). Conectar arduino por wifi con el módulo esp8266 esp01. Obtenido de http://repositorio.uisek.edu.ec/handle/123456789/3850

Martínez Lizares, G. V. (2020). Marketing de proximidad mediante aplicación móvil con dispositivos Beacon.

Mateo Castrillon, V. (2019). Sistema SCADA basado en un ambiente de programación open source.

Naranjo Heredia, S. J. (2021). Análisis del Stack Mern y su uso en las aplicaciones web basadas en servicios Rest. Obtenido de http://dspace.utb.edu.ec/handle/49000/9530

Ortiz-Garcés, Diego, P., & Pozo, A. (2020). Implementación de un sistema electrónico de seguridad portable para tarjetas bancarias contactless. Lousada.

Pérez Román, A. (2020). Comparación de rendimiento entre bases de datos Relacionales, NoSQL y Blockchain Comparación de rendimiento entre PostgreSQL, MongoDB y Kaleido.

Pérez Silva, O. (2019). Diseño de un plan de pruebas para la habilitación del servicio de red NB-IoT. Obtenido de https://repositorio.uchile.cl/handle/2250/173802

Remache, P. S. (2021). Diseño de Prototipo para control, monitoreo y analisis de radiacion solar y temperaturas en el noroccidente de la ciudad de Quito mediante sensores IoT.

Rendón Escudero, E. A. (2020). Desarrollo y despliegue de una plataforma web para el almacenamiento y administración de bioseñales.

Rodriguez Hernandez, J. R. (2021). Entendiendo github cli para gestionar organizaciones.

Samanpour, A. R. (2018). The Future of Machine Learning.

Sepúlveda, G. (2018). Implementación de RFID en un almacén logístico.

Setiawan, N. D. (2018). Otomasi Pencampur Nutrisi Hidroponik Sistem NTF (Nutrient Film Technique) Berbasis Arduino Mega 2560.

Siam, A. I.-Z.-S. (2021). Secure Health Monitoring Communication Systems Based on IoT and Cloud Computing for Medical Emergency Applications. Computational Intelligence and Neuroscience.

Siby, A. (2020). Android Hacking Using Msfvenom: Integrating NGROK.

Thomas B, B. (2022). Blue Charm Beacons.

Valderrama, J. &. (2020). ESP8266: Un microcontrolador para el Internet de las Cosas.

Vela Galindo, E. y. (2019). Desarrollo de una aplicación para el turismo en Madrid.

Vizcarra Cavero, S. I. (2019). Diseño e implementación de un sistema monitoreo inalambrico de bancos de baterías utilizando en Arduino Mega 2560.

5. ANEXOS

5. ANEXOS

ANEXO 1 MANUAL *ARDUTENFC WEBSITE* V1.0

Manual ArdUteNfc Website.pdf

ANEXO 2 MANUAL *ARDUTENFC TAG APP* V1.0

Manual de Usuario nfc.pdf

ANEXO 3 MANUAL *ARDUTENFC BEACON APP* V1.0

Manual de Usuario.pdf