UNIVERSIDAD UTE

FACULTAD DE CIENCIAS DE LA INGENIERÍA E INDUSTRIAS

CARRERA DE INGENIERÍA EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

DESARROLLO DE APLICACIONES MÓVILES PARA EL PROCESAMIENTO DE MÉTODOS Y TÉCNICAS DE DATA STREAMING PARA IoT

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO

DE INGENIERO EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

MARIO ALEJANDRO NARVÁEZ TAMAYO

DIRECTOR: ING. CIRO NAPOLEÓN SAGUAY CHAFLA

Quito, agosto de 2021

FORMULARIO DE REGISTRO BIBLIOGRÁFICO PROYECTO DE TITULACIÓN

DATOS DE CONTACTO	
CÉDULA DE IDENTIDAD:	1716793920
APELLIDO Y NOMBRES:	MARIO ALEJANDRO NARVÁEZ TAMAYO
DIRECCIÓN:	Urb. El Condado Calle V y Calle V1
EMAIL:	marinho.int@gmail.com
TELÉFONO FIJO:	607494
TELÉFONO MOVIL:	0999936644

DATOS DE LA OBRA		
TÍTULO:	DESARROLLO DE APLICACIONES MÓVILES PARA EL PROCESAMIENTO DE MÉTODOS Y TÉCNICAS DE DATA STREAMING PARA IoT.	
AUTOR O AUTORES:	MARIO ALEJANDRO NARVÁEZ TAMAYO	
FECHA DE ENTREGA DEL PROYECTO DE TITULACIÓN:	Agosto de 2021	
DIRECTOR DEL PROYECTO DE TITULACIÓN:	ING. CIRO NAPOLEÓN SAGUAY CHAFLA	
PROGRAMA	**PREGRADO** ☐	**POSGRADO** ☐
TÍTULO POR EL QUE OPTA:	INGENIERO EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN	
RESUMEN: Mínimo 250 palabras	En los últimos años se ha tenido un avance tecnológico realmente increíble, en el cual la tecnología se va incorporando en nuestra vida cotidiana, es así que se encuentra en todo tipo de artefactos electrónicos como sensores, GPS, cámaras, puestas, alarmas, luces, etc. Esta tecnología se conoce como IoT (Internet de las cosas). El uso e implementación de tecnología con	

dispositivos móviles ha ido creciendo y ha cobrado importancia en los últimos años, por lo cual la podemos integrar con IoT, por su alcance y portabilidad es perfecta para mostrar e interpretar los datos de IoT.

El internet de las cosas o IoT se encuentra en todo lado. Básicamente IoT son dispositivos que se conecta a internet los cuales pueden proveernos de cierta información. La mayoría de los dispositivos IoT son configurables para adaptarse a nuestras necesidades o también podremos construir nuestros propios dispositivos.

Gracias a estos avances en IoT se ha visto la necesidad de analizar, procesar, extraer valor a los datos y mostrarlos ya que estos se producen en tiempo real o también conocida como Data Streaming. La visualización de los datos en tiempo real es de bastante utilidad en las industrias debido a que pueden saber el estado de los dispositivos y los datos que envían estos en ese mismo instante. Al saber que sucede en el instante que pasa, puede ayudar a las empresas a resolver problemas inmediatamente, lo cual ayuda a mejorar la calidad de sus productos y optimizar sus costos.

Los datos que se obtienen a través de los dispositivos IoT son difíciles de interpretar par un ser humano, por lo cual se tiene que realizar una aplicación para poder capturarlos, procesarlos e interpretarlos a fin de que sean entendibles para un ser humano. Para mostrar e interpretarlos datos se ha optado por desarrollar una aplicación móvil. Gracias a los precios accesibles de los dispositivos móviles, más personas lo utilizan, por lo cual es muy factible el desarrollo de este proyecto en este entorno.

Por lo tanto, en este trabajo se pretende desarrollar una aplicación móvil que partiendo de datos generados en dispositivos IoT, permita el análisis, la visualización e interpretación de datos generados, para la toma de decisiones.

PALABRAS CLAVES:	**IoT, Móviles, Streaming, Procesamiento.**
ABSTRACT:	**In recent years there has been a truly incredible technological advance, in which technology is being incorporated into our daily lives, this is how it is found in all kinds of electronic devices such as sensors, GPS, cameras, settings, alarms, lights, etc. This technology is known as IoT (Internet of Things). The use and implementation of technology with mobile devices has been growing and has gained importance in recent years, so we can integrate it with IoT, because of its scope and portability it is perfect for displaying and interpreting IoT data.** **The internet of things or IoT is everywhere. Basically, IoT are devices that connect to the internet which can provide us with certain information. Most of the IoT devices are configurable to adapt to our needs or we can also build our own devices.** **Thanks to these advances in IoT, the need has been seen to analyze, process, extract value from the data and show it since these are produced in real time or also known as Data Streaming. The visualization of the data in real time is very useful in the industries because they can know the status of the devices and the data that they send at that very moment. By knowing what happens the moment it happens, you can help companies solve problems immediately, which helps improve the quality of their products and optimize their costs.** **The data obtained through IoT devices are difficult to interpret for a human being, so an application has to be made to be able to capture, process and interpret them so that they are understandable to a human being. To display and interpret the data, it has been decided to develop a mobile application. Thanks to the affordable prices of mobile devices, more people use it, which is why the development of this project in this environment is very feasible. Therefore, this work aims to develop a mobile application that, based on data generated in IoT devices, allows the analysis, visualization and interpretation of generated data, for decision-making.**
KEYWORDS	**IoT, Mobile, Streaming, Processing.**

Se autoriza la publicación de este Proyecto de Titulación en el Repositorio

Digital de la Institución.

MARIO ALEJANDRO NARVÁEZ TAMAYO
C.I. 1716793920

DECLARACIÓN Y AUTORIZACIÓN

Yo, **MARIO ALEJANDRO NARVÁEZ TAMAYO,** CI 1716793920 autor/a del proyecto titulado: **DESARROLLO DE APLICACIONES MÓVILES PARA EL PROCESAMIENTO DE MÉTODOS Y TÉCNICAS DE DATA STREAMING PARA IoT** previo a la obtención del título de **INGENIERO EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN** en la Universidad UTE.

- 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 graduación para que sea integrado al Sistema Nacional de información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.
- Autorizo a la BIBLIOTECA de la Universidad UTE a tener una copia del referido trabajo de graduación con el propósito de generar un Repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Quito, agosto de 2021

MARIO ALEJANDRO NARVÁEZ TAMAYO
C.I. 1716793920

CERTIFICACIÓN DEL TUTOR

Certifico que el presente trabajo que lleva por título "**DESARROLLO DE APLICACIONES MÓVILES PARA EL PROCESAMIENTO DE MÉTODOS Y TÉCNICAS DE DATA STREAMING PARA IoT.**", que, para aspirar al título de **Ingeniero en Informática y Ciencias de la Computación** fue desarrollado por **Mario Alejandro Narváez Tamayo** 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.

Ing. Ciro Saguay

DIRECTOR DEL TRABAJO

C.I. 0602692113

DECLARACIÓN JURAMENTADA DEL AUTOR

Yo **MARIO ALEJANDRO NARVÁEZ TAMAYO**, 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 este 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.

MARIO ALEJANDRO NARVÁEZ TAMAYO
C.I. 1716793920

ÍNDICE DE CONTENIDOS

Página

1. INTRODUCCIÓN 2

2. METODOLOGÍA 12

3. RESULTADOS Y DISCUSIÓN 17

3.1. HISTORIAS DE USUARIOS 17

3.2. HISTORIAS DE USUARIO TÉCNICAS 19

3.3. LIST PRODUCT BACKLOG 27

3.4. SPRINT PLANNING 29

3.5. ESTIMACIONES DE LOS SPRINTS 29

3.6. Sprin1 31

3.7. Sprint 2 37

3.7.1. Mutación para crear un usuario 37

3.7.2. Mutación crear dispositivo IoT 38

3.7.3. Mutación para editar el dispositivo IoT 39

3.7.4. Mutación para crear un evento de un dispositivo 39

3.8. Sprint 3 39

3.8.1. Query para mostrar los dispositivos IoT 39

3.8.2. Suscripción para el detalle de un dispositivo 40

3.9. Sprint 4 41

3.9.1. Lógica de la aplicación 42

3.9.2. Visualización de los dispositivos IoT 43

3.9.3. Detalle del dispositivo IoT 44

3.9.4. Cerrar sesión 45

3.10. Sprint 5 45

3.10.1. Construcción del dispositivo IoT 45

3.10.2 Programación del dispositivo IoT 46

4. CONCLUSIONES Y RECOMENDACIONES 49

1. INTRODUCCIÓN

1. INTRODUCCIÓN

En los últimos años ha existido un avance tecnológico realmente increíble, en el cual la tecnología se va incorporando en nuestra vida cotidiana, existe un sinnúmero de dispositivos que sé que se conectan a internet los cuales pueden proveernos de cierta información. Esta tecnología se conoce como IoT (Internet de las cosas), como le define (Liñán Colina, 2016), “la interconexión en red de objetos cotidianos, que a menudo están equipados con inteligencia ubicua”. El uso e implementación de tecnología con dispositivos móviles ha ido creciendo y ha cobrado importancia en los últimos años, por lo cual la podemos integrar con IoT, por su alcance y portabilidad es perfecta para mostrar e interpretar los datos de IoT.

Gracias a estos avances en IoT se ha visto la necesidad de analizar, procesar, extraer valor a los datos y mostrarlos ya que estos se producen en tiempo real o también conocida como data streaming, según (Amazon Web Services (AWS), 2020), una empresa dedicada a servicios en la nube establece que “son datos que se generan constantemente a partir de miles de fuentes de datos, que normalmente envían los registros de datos simultáneamente en conjuntos de tamaño pequeño”. La visualización de los datos en tiempo real resulta ser bastante útil en las industrias debido a que pueden saber el estado de los dispositivos y los datos que envían estos en ese mismo instante. Al saber que sucede en el instante que pasa, puede ayudar a las empresas a resolver problemas inmediatamente, ayudando a mejorar la calidad de sus productos y optimizar sus costos.

Gracias a los avances en IoT se ha visto la necesidad de mostrar los datos en tiempo real o también conocida como data streaming, según (Amazon Web Services (AWS), 2020) , “son datos de los cuales se puede extraer información valiosa para las empresas”. La visualización de los datos en tiempo real es de bastante utilidad en las industrias debido a que pueden saber el estado de los dispositivos y los datos que envían estos en ese mismo instante. Al saber que sucede en el instante que pasa puede ayudar a las empresas a resolver problemas inmediatamente, lo cual ayuda a mejorar la calidad de sus productos y optimizar sus costos.

Los datos que se obtienen a través de los dispositivos IoT son de difícil interpretación, frente a esto se requiere de la utilización de aplicaciones informáticas para poder recolectar, procesar e interpretar estos datos, con la finalidad que puedan ser entendibles para un ser humano.
En función de esta aseveración, el presente trabajo tiene como finalidad la creación de una aplicación informática que partiendo de datos generados en dispositivos IoT,

permita el análisis, la visualización e interpretación de datos generados, para la toma de decisiones.

El auge de los dispositivos de IoT significa que tenemos que recopilar, procesar y analizar órdenes de magnitud más de datos que nunca. A medida que los sensores y dispositivos se vuelven cada vez más omnipresentes, esta tendencia en los datos solo aumentará. Está transformando innumerables industrias: la atención médica ahora puede rastrear la salud de los pacientes en tiempo real e incluso brindar atención a pedido; la fabricación es capaz de comprender los detalles de las líneas de producción y predecir problemas antes de que sucedan; La industria automotriz está aprovechando los sensores no solo para la conducción autónoma, sino también para proporcionar información más profunda en tiempo real, lo que permite a la ingeniería solucionar problemas de manera preventiva antes de que los conductores los noten. Y esto es solo una muestra del potencial de los dispositivos de IoT, (Marcelo Alcaraz, 2020).

El Internet de las cosas (IoT) hace referencia a práctica constante de conectar todo tipo de dispositivos físicos al Internet, pudiendo ser cualquier tipo de objeto, desde refrigeradores, bombillas, dispositivos médicos, hasta dispositivos inteligentes e inclusive ciudades inteligentes que solo existen gracias al IoT.

Siendo más específicos, el término IoT hace alusión a los sistemas de dispositivos físicos que recogen y transportan datos mediante redes inalámbricas sin la participación humana. Por ejemplo, un "termostato inteligente" ("inteligente", por lo general, significa "IoT") recoge datos de la ubicación de su vehículo inteligente mientras conduce, y los utiliza para ajustar la temperatura de su casa antes de que llegue.
Un sistema de IoT tradicional, como el ejemplo anterior, trabaja enviando, recogiendo y analizando datos permanentemente en un ciclo de retroalimentación. Según el tipo de sistema de IoT, las personas o la inteligencia artificial y el aprendizaje automático (Machine Learning) pueden llevar a cabo el análisis casi en tiempo real; se debe mencionar que la inteligencia humana es natural, no existe algún mecanismo que alimenta al cerebro humano con algoritmos para actuar de cierta manera o hacer ciertas cosas, a diferencia de la inteligencia artificial, que busca emular al cerebro humano, sin embrago el proceso es artificial. Pensemos en un hogar inteligente, para predecir el instante óptimo para controlar el termostato antes de que llegue a casa, su sistema de IoT puede conectarse a la API de Google Maps para conseguir información sobre los patrones de tráfico en tiempo real de su área, y puede usar datos sobre sus hábitos de conducción que el automóvil recolecta a largo plazo. Además, las instituciones de servicios públicos pueden

analizar los datos de IoT que coleccionan de los clientes que tienen termostato inteligente, con la finalidad de optimizar el sistema a gran escala, (Red Hat, 2020).

Como lo afirma, (Marc, 2019) “Los dispositivos móviles están formados por un conjunto de componentes de hardware capaces de soportar una gran variedad de tecnologías inalámbricas, donde destaca uno o varios procesadores de altas prestaciones”. Esta capacidad de conexión es importante en especial con los dispositivos IoT porque estos se conectan a través de las tecnologías inalámbricas como: Bluetooth, wifi, RFID, entre otras. Otra ventaja de los dispositivos móviles es la gran diversidad existente tales como teléfonos inteligentes, tabletas, smartwatches, etc, cuya tecnología y funcionalidades nuevas mejoran continuamente. Dispositivos como los teléfonos pasaron de ser utilizados solamente para realizar llamadas o para enviar mensajes de texto, a disponer de funcionalidades avanzadas como: gestión de correo, reproductores multimedia, entre otras., convirtiéndose con esto en herramientas de trabajo, estudio y entretenimiento más usado y con mejor potabilidad del mundo.

Los datos de streaming son generados frecuentemente a partir de miles de orígenes de datos, que regularmente envían los registros de datos simultáneamente en conjuntos de tamaño pequeño. Los datos de streaming contienen varios tipos de datos, como archivos de registros generados por clientes que utilizan aplicaciones móviles o web, compras electrónicas, actividades de los jugadores en un juego, información de redes sociales, operaciones crediticias o servicios geoespaciales, así como telemetría de dispositivos conectados o instrumentación en centros de datos.
Los datos deben procesarse secuencial y gradualmente, registro por registro o en ventanas de tiempo graduales, y se emplean para una dilatada variedad de tipos de análisis, como: correlaciones, agregaciones, filtrado, muestreo entre otras. La información procedente del análisis aporta a las compañías claridad de muchos aspectos del negocio y de las actividades de los clientes, como el uso del servicio (para la medición/facturación), la actividad del servidor, los clics en un sitio web y la ubicación geográfica de dispositivos, personas y mercancías, y les permite responder con rapidez frente a cualquier situación que surja. Ejemplo, las compañías pueden supervisar los cambios en la opinión pública de sus marcas y productos al analizar constantemente las transmisiones de los medios sociales y responder rápidamente cuando sea necesario, (Amazon Web Services, 2019)**.**

La tecnología big data consta de datos que son extremadamente grandes o complejos, que no pueden manejarse con los métodos usuales de procesamiento.

También se conoce al big data por sus "tres V": volumen, variedad y velocidad. Los datos son valiosos, si se pueden preservar, procesar, vislumbrar y utilizar. El objetivo de big data es entregar información de forma inmediata, que mejore la toma de decisiones, (Red Hat, 2020).

Una de las fases del big data, es la recolección de datos, que posteriormente serán analizados. IoT nos proporciona un gran volumen de datos, estos pueden ser estructurados, no estructurados o semiestructurados. A pesar de que la tecnología Big Data describe la cantidad de datos, su importancia radica en el procesamiento de estos datos, con la finalidad de obtener conocimiento que permita tomar decisiones.
Según, (Pereira Villazón, 2019), la tecnología big data se conceptualiza habitualmente con tres Vs: volumen, velocidad y variedad, es el marco común para describir a los datos masivos. El volumen de datos hace referencia a la cantidad de datos que almacenamos, la velocidad esta encargada de realizar procesos matemáticos y estadísticos óptimos, con la finalidad que dichos datos sean analizados y mostrados en el menor tiempo posible, finalmente, la variedad se refiera a de donde provienen los datos y cómo están estructurados.

Hoy en día todas las organizaciones generan gran cantidad de datos, una fuente importante de datos, la constituyen los dispositivos IoT. A medida que estos datos aumentan exponencialmente, las aplicaciones IoT utilizan técnicas de minería de datos que analizan la enorme cantidad de datos para generar tendencias o patrones que se requieren para la toma de decisiones.

Los datos masivos generados por Internet de las cosas (IoT) se consideran de alto valor comercial, y los algoritmos de minería de datos se pueden aplicar a IoT para extraer información oculta de los datos. A medida que más y más dispositivos se conectan a IoT, se debe analizar un gran volumen de datos, los últimos algoritmos deben modificarse para aplicarlos a big data, (Frédéric Combaneyre, 2017).

Por otra parte, las técnicas de aprendizaje automático se están explotando recientemente dentro de los sistemas de IoT para aprovechar su potencial, mediante el uso de uso de algoritmos que explotan el aprendizaje automático en los sistemas de IoT. Dichos algoritmos brindan soluciones eficientes a los desafíos operativos básicos de IoT, como localización, agrupamiento, enrutamiento y agregación de datos, y aquellos que se enfocan en desafíos relacionados con el desempeño, como el control de la congestión, detección de fallas, gestión de recursos y seguridad, (Mansaf, Shakil, & Samiya, 2019).

Finalmente mencionar técnicas como: Inteligencia Artificial, Blockchain, Big Data que, aplicadas en diferentes áreas como la salud, agricultura, ciudades inteligentes, entre otras, ayudaran a optimizar el proceso de analítica de datos generados por dispositivos IoT.

Python es un lenguaje de programación interpretado cuya filosofía principal es que sea legible para cualquier persona, tiene una serie de características que lo hacen muy particular y que, sin duda, le aportan muchas ventajas:

- ***Es totalmente gratuito***, es un lenguaje open source.
- ***Está respaldado por una enorme comunidad***, al ser un lenguaje gratuito, continuamente se están desarrollando nuevas librerías y aplicaciones.
- ***Es un lenguaje multiparadigma***, combina propiedades de diferentes paradigmas de programación.
- ***Sus aplicaciones no se limitan a un área en concreto***, permite utilizarlo en campos aparentemente tan disímiles como el desarrollo de aplicaciones web o la inteligencia artificial, entre otros.
- ***Python es apto para todas las plataformas***, se puede ejecutarlo en diferentes sistemas operativos como Windows o Linux, con el simple uso del intérprete correspondiente

El lenguaje de programación Python es largamente utilizado por compañías de todo el mundo para el desarrollo de aplicaciones web, analítica de datos, automatizar operaciones y crear soluciones fiables y escalables. Ejemplo de compañías tecnológicas, se encuentran Google, Uber, Netflix y Facebook, llevan años utilizando este lenguaje de programación y construyendo su infraestructura tecnológica basándose en él, (Universidad de Sevilla, 2020).

Dentro de las bases de datos NoSQL, seguramente una de las más utilizadas es MongoDB. Esta base de datos tiene un concepto distinto a la de las bases de datos tradicionales. Esta base de datos resulta útil cuando queremos almacenar información de: sensores, redes sociales, páginas web etc. MongoDB es una base de datos orientada a documentos, significa que, en lugar de guardar los datos en registros, guarda los datos en documentos. Tales documentos son almacenados en BSON, que constituye en una representación binaria de JSON.

El servicio de base de datos en la nube más innovador del mercado, con una distribución y movilidad de datos inigualables en AWS, Azure y Google Cloud, automatización integrada para la optimización de recursos y cargas de trabajo, y mucho más.

MongoDB Atlas es el servicio de base de datos en la nube, su implementación esta administrada por las empresas: AWS, Google Cloud y Azure, que garantizan la disponibilidad, escalabilidad y cumplimiento de los estándares de privacidad y seguridad de datos más exigentes, (Mongo DB, 2020).

Amazon Web Services, conocida también como AWS, es un conjunto de herramientas y servicios de cloud computing de Amazon. Una característica importante de esta herramienta es su madurez en el servicio que ofrece, lo cual les hace una herramienta muy diferente frente a sus competidores; además de ofrecer un amplio abanico de herramientas disponibles: Cloud computing, creación de redes virtuales, inteligencia de negocios, internet de las cosas entre otros, (Amazon Web Services, 2019).

Lanzado en 2006, Amazon Web Services (AWS) es una colección de servicios de computación en la nube pública, que en conjunto constituyen una de las más importantes plataformas de computación en la nube, ofertadas a través de Internet por Amazon.com. Es utilizada en muchas aplicaciones populares como: Dropbox, Foursquare, Pinterest, entre otras. Sus principales competidores en este ámbito esta Microsoft Azure y Google Cloud Platform.

Amazon Web Services ofrece servicios en línea para otros sitios o aplicaciones web del lado del cliente. Gran parte de estos servicios no están expuestos directamente a los usuarios finales, sino que ofrecen una funcionalidad que otros desarrolladores puedan utilizar en sus aplicaciones, se puede acceder a este servicio, a través de HTTP, utilizando protocolos REST y SOAP, (Amazon Web Services, 2019).

La aparición de Node.js ha significado toda una revolución en el mundo de JavaScript. Herramienta elaborada en base del motor Chrome V8, Node.js es un entorno de código abierto cuyo el desarrollo de aplicaciones JavaScript, se lo realiza del lado del servidor. Esto permite que, una vez desarrolladas las aplicaciones, estas puedan ejecutarse en diferentes sistemas operativos como OS X, Microsoft Windows y Linux. Esta herramienta también proporciona una amplia biblioteca con módulos JavaScript, lo cual simplifica el desarrollo de aplicaciones web usando este framework en gran medida.

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor, basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor de Google. Fue desarrollado con el enfoque de ser útil en la creación de programas de red altamente escalables, como, por ejemplo, servidores web.

Permite a los desarrolladores escribir herramientas de línea de comandos y scripts del lado del servidor fuera de un navegador, (Node JS, 2018).

Para dispositivos de Internet de las Cosas (IoT) se hace indispensable la conexión a Internet. La conexión a Internet admite que los dispositivos trabajen entre ellos y con servicios backend. La conectividad es un conjunto de protocolos para una red en Internet, los más utilizados son HTTPS, Websockets y Message Queue Telemetry Transport (MQTT) cada uno de estos tiene diferente propósito y usabilidad. El transporte de telemetría de mensaje en cola (MQTT) es un protocolo utilizado para la comunicación dentro de un entorno IoT.

MQTT originalmente, fue desarrollado e inventado por IBM a finales de los años 90, su utilidad inicialmente era conectar sensores de los oleoductos con satélites. Tal cual su nombre, es un protocolo de mensajería que soporta la comunicación asíncrona entre las partes. Un protocolo de mensajería asíncrona disgrega al emisor y al receptor de los mensajes tanto en espacio como en tiempo, y, por lo tanto, es escalable en entornos de red no confiables. Contrariamente a su nombre, no tiene ninguna relación con las colas de mensajería, al contrario, utiliza un modelo de publicación y suscripción. A finales del 2014, se convirtió oficialmente en un estándar abierto, y se soporta en lenguajes de programación populares mediante la utilización de diferentes implementaciones de código abierto.

En IoT se usa este protocolo porque es considerablemente ligero en cuanto al transporte de datos, esto lo hace funcional porque IoT maneja una cantidad muy grande de datos y mientras más ligero es el transporte, alta es la conectividad. Con esta conectividad se puede enviar los datos a un servidor y este se encargará de mostrar estos datos a los clientes. Al tener una conexión rápida y ligera se reduciría al mínimo la latencia o el tiempo de respuesta; lo cual ayudará a tener muchos más dispositivos conectados y reducirá la cantidad de recursos utilizados por el servidor, (IBM Developer, 2017).

Una vez que se ha determinado el marco teórico que ha sido utilizado para el presente trabajo, a continuación, se muestra la metodología aplicada en la ejecución de las actividades de la presente investigación.

Como en todo proyecto se tiene que alcanzar un objetivo por lo cual necesitamos una metodología que ayude a llevar a cabo dicho objetivo. La metodología Scrum es la utilizada para el desarrollo el presente trabajo, según (Dimes, 2015) . Scrum es un macro de referencia para crear software complejo y entregarlo a tiempo de una forma mucho más sencilla. Debido a la facilidad de uso, y a los resultados obtenidos al aplicarla, tiene una gran aceptación y usabilidad en el campo laboral.

Esta metodología está estructurada por un equipo con diferentes roles: Product Owner, Scrum Máster y el Developer Team.

El núcleo de Scrum es un Sprint, es un intervalo determinado durante el cual se crea un prototipo de producto "Hecho o Terminado" utilizable, y potencialmente entregable. A lo largo del desarrollo hay Sprints consecutivos de duración constante. Al igual que los proyectos, los Sprint se utilizan para lograr algo.

Scrum Máster: Este rol conoce muy bien la metodología de tal manera que está instruido de cómo funcionan todos los procesos, es un guía debido a que enseña y orienta al resto del equipo, no obstante, no se constituye en el líder del grupo, es flexible al momento de debatir sobre un cambio y está encargado de realizar reportes en las reuniones.

Product Owner: Este rol decide la prioridad y conoce bien del negocio, permite detallar los requerimientos de manera precisa y clara.

Development Team: Es el equipo que se encarga de entregar el producto, se recomienda constituir un equipo pequeño de 3 a 9 personas, cuyos integrantes deben alcanzar a tener un vínculo fuerte como equipo.

Como lo mencionamos anteriormente, scrum está conformado por sprints los cuales cumplen un ciclo que tiene 4 fases: sprint, daily, revisión del sprint y retrospectiva del sprint. En la planeación se determinan todas las tareas que se van a realizar en este, normalmente las tareas duran de 1 a 8 horas, si una tarea durará más de 8 horas esta se la divide. El daily es una revisión rápida de lo que se avanzó en el día y ver si hay algún bloqueo en alguna tarea, esto no dura más de 5 minutos. En la revisión del sprint se muestra una demo de todo lo que se realizó. Y en la retrospectiva se realiza una revisión de lo que se hizo bien y mal durante el sprint y determinar qué es lo que se tiene que seguir haciendo, lo que se tiene que mejorar y lo que se tiene que dejar de hacer. En la Figura 1 se muestra el ciclo de un sprint en scrum.

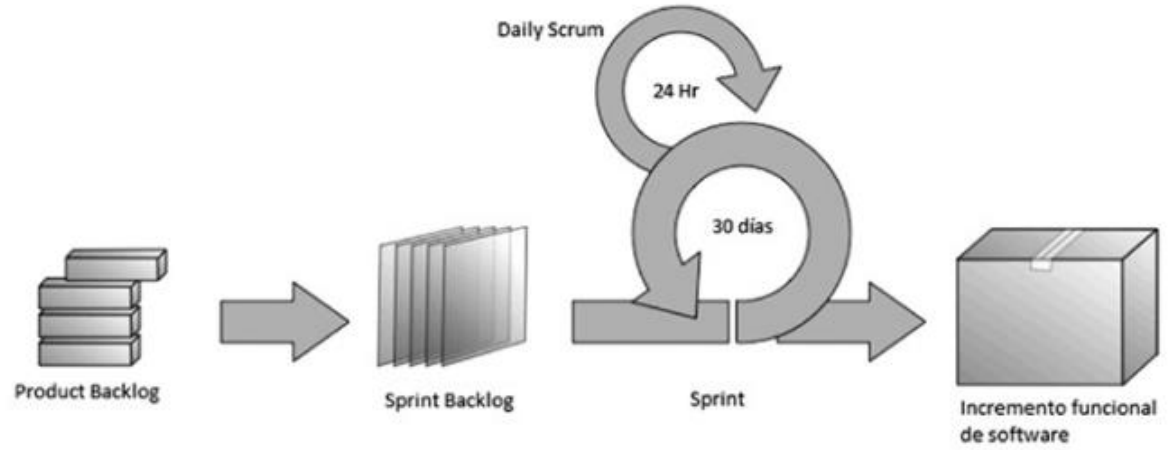

Figura 1. Fases de un sprint (Andrés Navarro Cadavid, Juan Daniel Fernández Martínez, Jonathan Morales Vélez, 2013)

Debido a que la metodología scrum es muy fácil de utilizar y adaptable a cualquier tipo de proyecto, se decide emplearla en el presente trabajo. Con scrum se puede tener bastante claro como el proyecto está avanzando, tener mayor control sobre este y poder llevar a cabo en el tiempo establecido este proyecto.

Par este proyecto se tiene como objetivo general diseñar y desarrollar una aplicación móvil que permita procesar, transmitir y visualizar datos recibidos desde dispositivos IoT. Los objetivos específicos de este proyecto son:

- Construir y configurar un prototipo de dispositivo IoT.
- Analizar las tecnologías disponibles para el tratamiento de data Streaming
- Diseñar y desarrollar un servicio que permita almacenar y procesar la data recibida desde dispositivos IoT.
- Diseñar y desarrollar una aplicación móvil que permita procesar, analizar y visualizar los datos.

2. METODOLOGÍA

2. METODOLOGÍA

Todo proyecto debe tener una correcta planificación lo que permitirá un desarrollo fluido y evitara conflictos en el proceso, por esta razón se decidió utilizar la metodología SCRUM. Esta metodología nos ayuda a organizar las tareas a realizar para cumplir con los objetivos del proyecto, esto se la hace atreves de historias de usuario, (Ágiles, Pontificia Universidad Católica del Perú; Ágiles, Pontificia Universidad Católica del Perú; Ágiles, Pontificia Universidad Católica del Perú; Ágiles, Pontificia Universidad Católica del Perú; Ágiles, Pontificia Universidad Católica del Perú). El tiempo es un factor importante en el momento de realizar el un proyecto con SCRUM podremos entregar el proyectó en el tiempo indicado médiate la planificación del sprint.

Como ya se mencionó este proyecto se lo va a realizar con SCRUM. El primer paso en este proyecto es realizarán las historias de usuario las cuales ayudan a identificar los objetivos y requerimientos del proyecto. El segundo paso es planificar las tareas que se van a realizar para cumplir con los objetivos y requerimientos del proyecto, a las culés se las va a organizar por Sprint's. Y por último ser realizan las tareas de los Sprint's.

Para las historias de usuarios se analizaron los objetivos y requerimientos del proyecto, en estas se especifica a detalle lo que se requiere para poder crear las tares necesarias y establecer los límites del proyecto. Al realizar este paso se obtuvo una mejor perspectiva de que se realizara en este proyecto. En cuanto al desarrollo de la aplicación móvil se la va a realizar multiplataforma para abarcar a un público más grande. En cuanto a la trasmisión de los datos se lo realizara a través de un servidor en la nube, por su costo y fácil acceso. El almacenamiento de los datos se los va a alojar en una base de datos NoSQL porque tiene mayor flexibilidad es más rápido y tiene mayor compatibilidad con los dispositivos IoT. El análisis, procesamiento y tratamiento de los datos se lo va a realizar con Python gracias a que este lenguaje de programación tiene varias librerías las cuales ayudan con este proceso. Y en cuanto a la construcción del dispositivo IoT se va a utilizar un Arduino Uno porque es el más fácil de conseguir y utilizar.

Para el segundo paso de este proyecto se realiza un Sprint planning, esto ayuda a organizar y crear las tareas que se tienen que hacer y en qué tiempo se las tiene que realizar. En base a las historias de usuario se planifican las tareas para cumplir con los objetivos y requerimientos del proyecto, al momento de crear la tarea se coloca un título y una breve descripción de lo que se tiene que hacer. Para la organización de las tareas se los divide en distintos Sprint's en este caso cada Sprint

tiene un periodo de 10 días, al final de cada Sprint se realiza una retrospectiva de las tareas que se realizaron y las que no se pudieron concluir. También cada tarea que se planifico debe tener un tiempo, este es una estimación de lo que se cree que se va a demorar en concluir una tarea, lo recomendable es siempre estimar con más tiempo de lo que se espera por si existe alguna complicación. Los Sprint y las tareas que se tienen que realizaría están con más detalle en la sección de resultados y discusiones.

Al concluir el Sprint planning se crearon diversas tareas divididas en 5 Sprint's con un total de 50 días para concluir con el proyecto. El primer Sprint se realizan todas las configuraciones en nuestra maquina y en los servicios en la nube para poder comenzar con el proyecto, esto se logró en el tiempo estimado. El segundo Sprint se realizó el servidor el cual tiene que alojar los datos, para esto se ingresaron datos de prueba, este Sprint también cumplió con los tiempos. En el tercer Sprint se continuo con el servidor, en este se realizó el análisis de los datos y la transferencia de datos en tiempo real con el protocolo MQTT, este no se cumplió con el tiempo, se demo un día de más por hubo problemas con MQTT. En el cuarto Sprint se realizó la aplicación móvil con React Native, en este se trabajó en el login, logout, registro de usuario y en listar los dispositivos IoT ligados al usuario, se lo realizo con el tiempo estimado. Y finalmente en el Sprint cinco se terminó la aplicación móvil mostrando los gráficos en tiempo real y también se construyó el dispositivo IoT, este sprint también se lo realizo con el tiempo estimado. En total se demoró 51 días en finalizar este proyecto.

Para que este proyecto funcione adecuadamente se implementaron varias tecnologías durante los Sprint's. Entre estas tecnologías se encuentran algunos servicios en la nube de AWS para facilitar el desarrollo del proyecto. Una base de datos NoSQL para poder almacenar la información de los dispositivos IoT. Un servidor en Python para analizar y enviara los datos en tiempo real. Una aplicación en React Native para mostrar los datos en un dispositivo móvil sea este IOS o Android. También se cero un dispositivo IoT con un Arduino Uno, el cual envía los datos generados al servidor.

Los servicios de AWS que se usaron en este proyecto fueron AWS API Gateway, AWS Lambda y AWS Cognito. El servicio de AWS API Gateway nos permite exponer nuestra API desde un enlace HTTPS y de esta manera podemos consumir nuestro back-end desde cualquier aplicación o dispositivo. El servicio de AWS Lambda se complementa con el de API Gateway porque cuando una petición es realiza en AWS API Gateway esta llama a la función lambda la cual te devuelve un resultado y este resultado es enviado al usuario por medio de AWS API Gateway.

En cuanto al servicio de Cognito se lo escogió para manejar los usuarios de este proyecto, con Cognito se evita programar todo lo que tiene que ver con los usuarios como la codificación y decodificación de las claves, el alojamiento de la información, gracias a esto se redujo el tiempo de desarrollo.

En este proyecto también se utilizó una base de datos NoSQL, en este caso MongoDB, para el almacenamiento de los eventos del dispositivo IoT. Se escogió una base de datos NoSQL porque esta es mucho más flexible que una base de datos SQL y por su velocidad esta optimizada cuando se trata de grandes cantidades de datos como lo eventos en los dispositivos IoT. Para este proyecto se decidió utilizar MongoDB porque es la base de datos NoSQL más popular, con una comunidad bastante amplia la ayudo en a cualquier persona al instante. Para este proyecto debe tener una base de datos disponible 24/7 por lo que se decidió utilizar Mongo Atlas, un servicio en la nube.

Hasta el momento ya se tiene claro los servicios que se van a utilizar para este proyecto, ahora continuando con el Sprint planning se tiene que desarrollar el servidor. Este tiene que enviar la información de los dispositivos IoT disponible al igual que almacenar los datos generados por estos. Los datos generados por el dispositivo IoT se los tiene que analizar para enviar los datos ya tratados al cliente. También tiene que enviar los datos en tiempo real por medio del protocolo MQTT. Y por supuesto esto tiene que estar asegurado para que solo las personas registradas puedan acceder a este servicio.

Para el servidor vamos a utilizar el lenguaje de programación Python porque es el mejor en cuanto al realzar análisis de datos. Con el lenguaje de programación Python se creó una API GQL la cual realiza varias peticiones para que el usuario las pueda consumir. Este servicio devuelve todos los dispositivos IoT ligados a un usuario también devuelve un análisis de los datos generados por un dispositivo IoT en tiempo real por medio del protocolo MQTT. Todos estos datos son guardados por el servidor en Mongo Atlas para ser analizados. En cuando a la seguridad se utilizó Cognito para asegurar el servicio que está en la nube, de esta manera solo los usuarios registrados pueden entrar a este. Y por último este programa es subido a la nube por medio de una función lambda. En pocas palabras lo que se hace en este paso es unir todos los servicios que se definieron en una función lambda.

La siguiente parte que se planifico en el esprinte planning es la creación de la aplicación móvil la cual tiene que mostrar todos los datos en un formato que el ser humano los pueda entender. Para este se utilizó React Native, con este se puede crear una aplicación móvil tanto en Android como en IOS sin la necesidad de utilizar diferentes entornos de programación. React Native también es soportado por una gran comunidad lo cual facilita el trabajo debido a que si existiera algún problema en el desarrollo es muy probable que exista una solución en los foros. También para probar en un dispositivo IOS se requiere una computadora de la marca Apple, esto es debido a las políticas de uso de Apple.

Para la creación del dispositivo IoT se utilizó un Arduino uno, un ethernet Shield, un sensor DHT22, resistencias de 10k y cables de conexión. Obtener las piezas y construir el dispositivo IoT es el primer, después de construir el dispositivo IoT se lo tienen que programar, para esto se decidió utilizar jonny-five por su compatibilidad con la API. Como ya se ha descrito este proyecto está conformado por varias tecnologías diferentes las cuales al unirse hace que todo funcione como se lo espera. En el siguiente grafico se observa cómo está estructurado todo este proyecto.

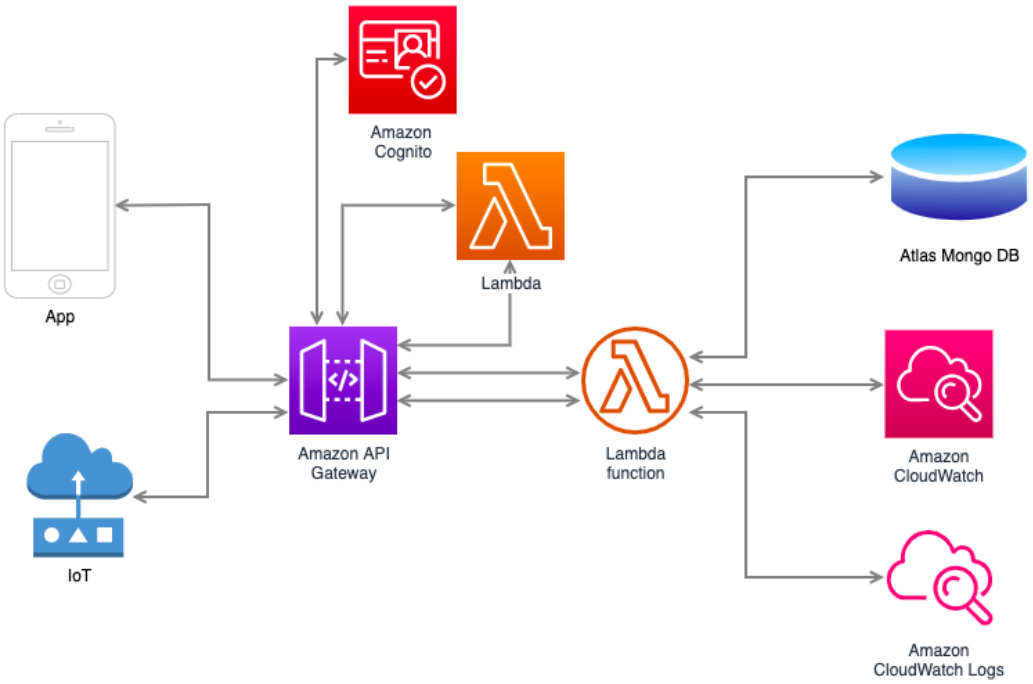

Figura 2. Estructura del proyecto

3. RESULTADOS Y DISCUSIÓN

3. RESULTADOS Y DISCUSIÓN

A continuación, se mencionan los resultados obtenidos en el presente proyecto de titulación, en este también se incluirá la arquitectura utilizada en el desarrollo del proyecto, un resumen de las tecnologías implementadas y de cómo estas respondieron ante el marco utilizado para así poner en evidencia el correcto funcionamiento de este.

Para obtener como resultado una aplicación satisfactoria se pasó por un proceso y una planeación. Se utilizo la metodología SCRUM para facilitar el desarrollo, dar un orden y un proceso por el cual este proyecto nos dio diferentes resultados en cada etapa del sprint en la metodología SCRUM.**HISTORIAS DE USUARIOS**

Las historias de usuario son fundamentales para la metodología SCRUM debido a que con estas se establece el alcance y tareas que se tienen que hacer para completar el proyecto. Estas historias de usuario tienen un ID de la historia para poder diferenciarlas, una descripción de lo que se tiene que hacer y los requisitos de aceptación, las historias se muestran en la Tabla 1 siguiente:

Tabla 1. Historias de usuarios

Historias de Usuarios				
ID	Usuario	Funcionamiento	Evento	Requisitos de aceptación
H1	Usuario general	Tener una pantalla de inicio de sesión	Al abrir la aplicación.	Se requieren visualizar los inputs para ingresar el usuario y contraseña, un botón de login y otro de registrarse.
H2	Usuario Registrado	Actividad que permita al usuario iniciar sesión con su cuenta	Clic en el Botón de login	Permitir el ingreso a la aplicación el 99% de las veces.
H3	Usuario general	Actividad que permita registrar un usuario	Llenar el formulario de registro y Clic en el botón de registrar	Al registrarse enviar un email de confirmación

H4	Usuario general	Listar todos los dispositivos IoT registrados por usuario.	Al iniciar la sesión	Lista de todos los dispositivos IoT del usuario.
H5	Usuario general	Modal para Editar y agregar un dispositivo IoT	Al presionar en el botón de editar o crear dispositivo IoT	Mostrar un modal con el formulario para editar o crear un dispositivo IoT
H6	Usuario general	Editar y agregar un dispositivo IoT	Clic en update o create	Se tiene que editar o agregar un dispositivo en el 99% de las veces. Cuando se agregue o edite la lista se tiene que actualizar.
H7	Usuario general	La capacidad de analizar los datos en tiempo real. Los datos se actualizan con el transcurso del tiempo.	Clic en details, se mostrar una nueva pantalla.	El análisis de los datos no tiene que tardar más de 5 segundos. Y tiene que estar disponible el 99% de las veces
H8	Usuario general	Datos provenientes del servidor tienen que ser legibles para el ser humano.	Visualización de la pantalla de detalles del dispositivo IoT.	Los datos que se obtiene del servidor tienen que estar representados mediante gráficos para su comprensión.
H9	Usuario general	Solo los usuarios pueden acceder al servidor.	Al momento de hacer cualquier petición a excepción de registrarse.	El servidor solo enviara los datos cuando el usuario haya ingresado satisfactoriamente.
H10	Usuario general	Guardar sesión	Al ingresar en la aplicación.	La primera pantalla que aparecerá es la lista de dispositivos IoT, si y solo si el usuario ya ingreso a su sesión.

H11	Usuario general	Cerrar sesión	Al presionar en “Logout”	La aplicación eliminara la referencia de sesión de usuario y le enviara a la pantalla de login.

3.2. HISTORIAS DE USUARIO TÉCNICAS

Una vez desarrolladas las historias de usuario se tiene que realizar las historias de usuario técnicas tal como se puede apreciar en la Tabla 2, estas contienen especificaciones más detalladas de cómo se van a realizar las tareas. En esta sección solo se encuentran las especificaciones técnicas del proyecto mas no como este fue creado.

Tabla 2. Historias de usuarios técnicas

Historia de usuario 1	
Nombre:	Estructura del sistema
Descripción:	Configuraciones previas al sistema
Prioridad:	Alta
Riesgo:	Alta
Requerimientos: • Instalación de IDE seleccionado para el desarrollo. • Crear una ambiente lambda para alojar el servidor, en AWS. • Crear un ambiente para alojar la base de datos, en MongoDB atlas. • Crear un servicio para la autenticación del usuario, en AWS Cognito.	
Historia de usuario 2	
Nombre:	Configuración de lint
Descripción:	Crear reglas para que el código sea legible y evitar errores de compilación.
Prioridad del negocio:	Baja
Riesgo:	Baja

Requerimientos • Con un comando se arregla el estilo del código y muestra si hay algo que arreglar.	
Historia de usuario 3	
Nombre:	Mutación crear usuario
Descripción:	El servidor con capacidad para crear un usuario por medio de una mutación.
Prioridad del negocio:	Alta
Riesgo:	Medio
Requerimientos: • Se tiene que crear un usuario con un email único. • Si el email ya existe tiene que enviar un error de que el email ya está ocupado.	
Historia de usuario 4	
Nombre:	Mutación crear dispositivo IoT
Descripción:	El servidor con capacidad para almacenar un dispositivo IoT por medio de una mutación.
Prioridad del negocio:	Alta
Riesgo:	Medio
Requerimientos: • El servidor tiene que almacenar un dispositivo IoT con un nombre único por usuario. • Si el nombre ya existe tiene que enviar un error de que el nombre ya está ocupado.	
Historia de usuario 5	

Nombre:	Mutación editar dispositivo IoT
Descripción:	El servidor con capacidad para editar los datos generales del dispositivo IoT.
Prioridad del negocio:	Alta
Riesgo:	Medio
Requerimientos: • El servidor edita la información de un dispositivo IoT que esta almacenado en la base de datos.	
Historia de usuario 6	
Nombre:	Mutación agregar datos
Descripción:	El servidor con capacidad para agregar datos del dispositivo IoT
Prioridad del negocio:	Alta
Riesgo:	Alta
Requerimientos: • El servidor tiene que guardar todos los datos enviados por el dispositivo IoT en la base de datos.	
Historia de usuario 7	
Nombre:	Query mostrar dispositivos
Descripción:	El servidor con capacidad para enviar una lista de los dispositivos IoT por usuario.
Prioridad del negocio:	Alta
Riesgo:	Media

Requerimientos: • El servidor tiene que enviar una lista de los dispositivos IoT. • La lista que se envía tiene que ser solo del usuario actual.
Historia de usuario 8
Nombre: Suscripción detalle dispositivo
Descripción: El Servidor con capacidad para enviar la información procesada del dispositivo en tiempo real.
Prioridad del negocio: Alta
Riesgo: Alta
Requerimientos: • Cada vez que el dispositivo IoT envié información al servidor y el usuario esta suscripto se actualizara la información de este.
Historia de usuario 9
Nombre: Login de la aplicación
Descripción: Realizar la pantalla y agregar la funcionalidad para el login
Prioridad del negocio: Alta
Riesgo: Alta
Requerimientos: • Realizar la parte estética, que tenga un look aceptable. • Conectar el login con el servicio de AWS Cognito.

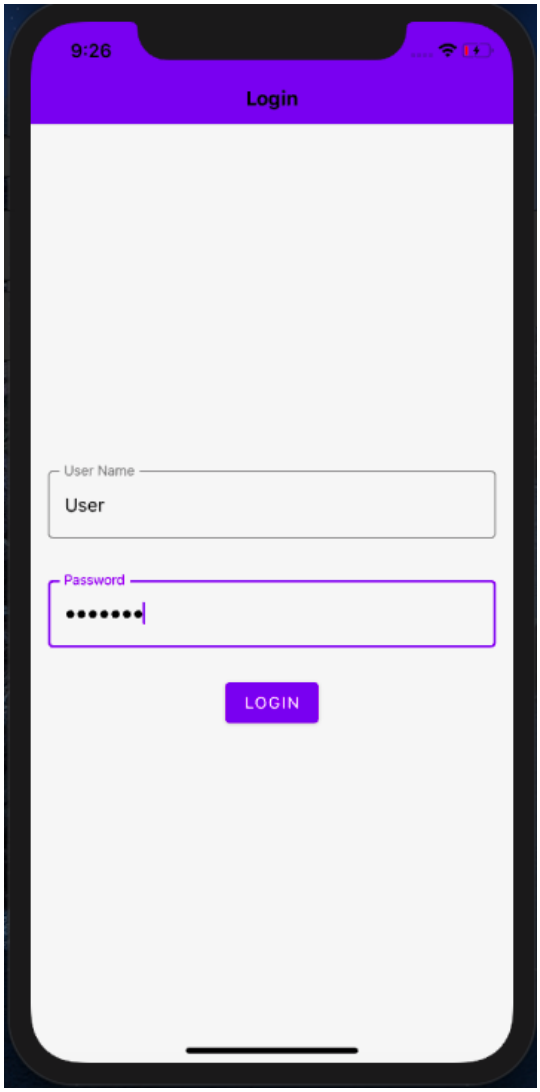

Figura 3. Pantalla de inicio.

Historia de usuario 10	
Nombre:	Visualización de los dispositivos IoT
Descripción:	Realizar una pantalla en donde se muestran todos los dispositivos IoT y poder editarlos.
Prioridad del negocio:	Alta
Riesgo:	Alta

Requerimientos:

- Realizar la parte estética, que tenga un look aceptable.
- Implementar la query para mostrar los dispositivos IoT.
- Implementar la mutación para editar los dispositivos IoT.
- Implementar la mutación para crear un dispositivo IoT.

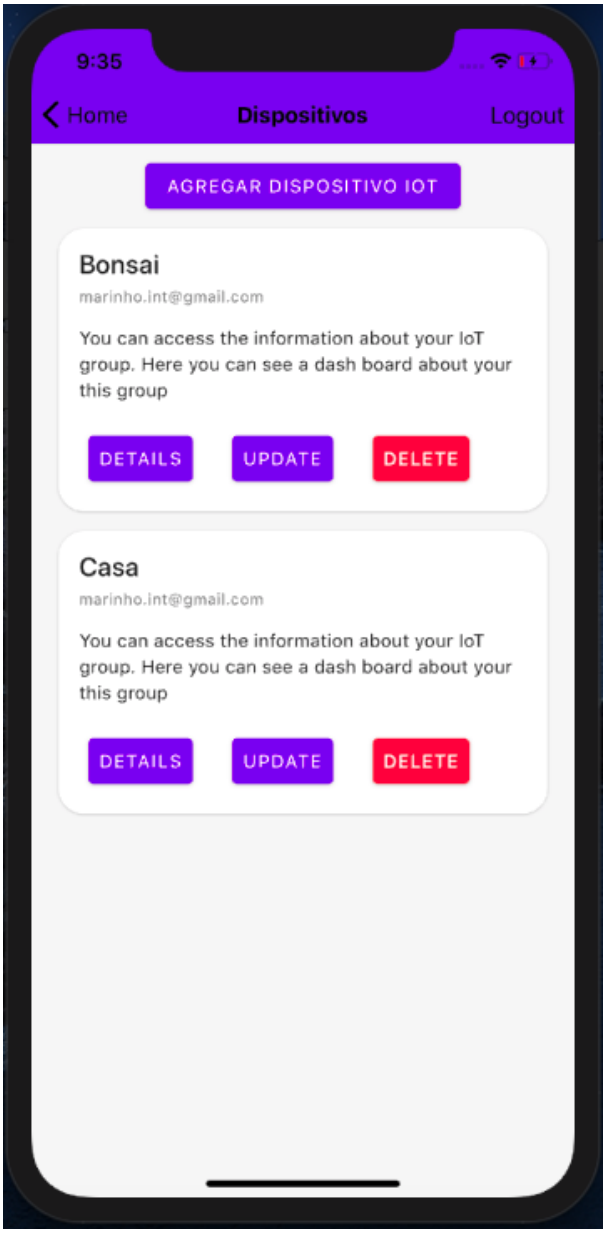

Figura 4. Lista de dispositivos IoT.

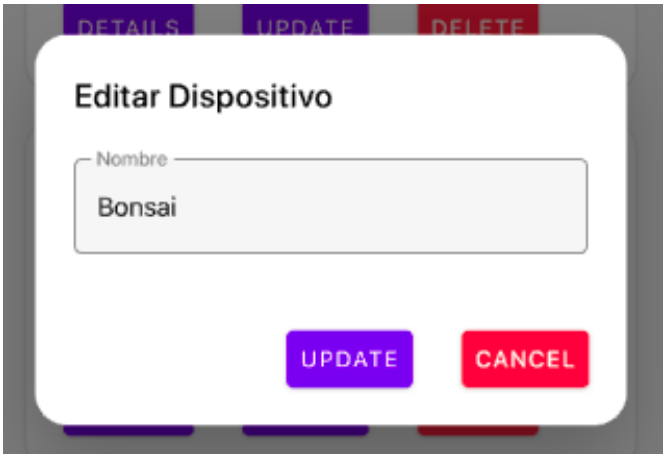

Figura 5. Creación y edición de dispositivo IoT.

Historia de usuario 11	
Nombre:	Detalle del dispositivo IoT seleccionado
Descripción:	Cuando se seleccione un dispositivo IoT, se muestra una pantalla con con un gráfico de los datos actuales y con otro mostrando las estadísticas.
Prioridad del negocio:	Alta
Riesgo:	Alta

Requerimientos:

- Realizar la parte estética, que tenga un look aceptable.
- Implementar la suscripción para mostrar los datos en tiempo real.

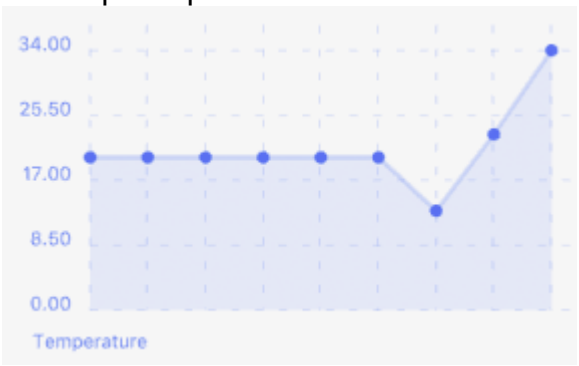

Figura 6. Gráfica del estado actual del dispositivo IoT

Historia de usuario 12	
Nombre:	Cerrar sesión
Descripción:	Cuando el usuario de clic en cerrar sesión esté se desconectará y volverá a

	la pantalla de inicio.
Prioridad del negocio:	Media
Riesgo:	Media

Requerimientos

- Un botón que esté en la parte superior derecha al cual de dos clic y nos lleva al inicio de la aplicación.

Figura 7. Botón para cerrar la sesión.

Historia de usuario 13	
Nombre:	Dispositivo IoT
Descripción:	Crear un dispositivo IoT el cual pueda enviar datos al servidor.
Prioridad del negocio:	Media
Riesgo:	Media

Requerimientos

- Construcción del dispositivo IoT, con acceso a internet y un sensor que envié información.

Historia de usuario 14	
Nombre:	Programar el dispositivo IoT
Descripción:	Programar el dispositivo IoT para que envié la información al servidor.
Prioridad del negocio:	Alta
Riesgo:	Alta

Requerimientos
- Enviar la información al servidor cada 5 segundos para que esta pueda ser procesada analizada y visualizada.

3.3. LIST PRODUCT BACKLOG

En SCRUM el Product Backlog es una lista ordenada de todas las tareas que se necesitan para completar el proyecto. Esta lista da disponibilidad y prioridad de las tareas pendientes del proyecto, a continuación, se muestra el detalle de los Product Backlog:

Tabla 3. Product Backlog Detallado

ID Historia de usuario	Requerimientos	ID Tarea	Tareas
H1	Configuraciones previas al sistema	H1.1	Instalación de IDE seleccionado para el desarrollo.
		H1.2	Crear una ambiente lambda para alojar el servidor, en AWS.
		H1.3	Crear un ambiente para alojar la base de datos, en MongoDB atlas.
		H1.4	Crear un servicio para la autenticación del usuario, en AWS Cognito
H2	Configuración de lint	H2.1	Determinar las reglas de lint.
		H2.2	Crear el comando para formatear el código.
H3	Mutación crear usuario	H3.3	Definir el esquema de usuarios en graphql.
		H3.2	Crear una mutación con graphql.
		H3.3	Conectar la creación con el servicio de cogito de AWS.
H4	Mutación crear dispositivo IoT	H4.1	Definir el esquema de dispositivos IoT en graphql.

		H4.2	Crear la mutación con graphql.
		H4.3	Conectar con la base de datos.
H5	Mutación editar dispositivo IoT	H5.3	Crear la mutación con grapqhl.
		H5.2	Conectar con la base de datos.
H6	Mutación Agregar data	H6.1	Definir el esquema de los datos de los dispositivos IoT
		H6.2	Crear la mutación con graphql.
		H6.3	Conectar con la base de datos.
H7	Query mostrar dispositivos	H7.1	Crear la query con graphql.
		H7.2	Conectar con la base de datos.
H8	Suscripción detalle dispositivo	H8.1	Crear la suscripción con graphql.
		H8.2	Conectar con un websoket.
		H8.3	Conectar con la base de datos.
H9	Login de la aplicación	H9.1	Interfaz con botones e inputs.
		H9.2	Conectar con el servicio de AWS Cognito
H10	Visualización de los dispositivos IoT	H10.1	Interfaz con lista, botones y modales
		H10.2	Obtener los datos de la query para visualizarlos.
		H10.3	Formulario para crear dispositivo IoT con su respectiva mutación.
		H10.4	Formulario para actualizar dispositivo IoT con su respectiva mutación.
H11	Detalle del dispositivo IoT seleccionado.	H11.1	Interfaz con botones y gráficos
		H11.2	Obtener los datos de la suscripción.
H12	Cerrar sesión	H12.1	Interfaz con el botón.

		H12.2	Conectar con el servicio de AWS Cognito.
H13	Dispositivo IoT	H13.1	Construir el dispositivo IoT con los materiales.
H14	Programar el dispositivo IoT	H14.1	Programar el dispositivo IoT en js con johnny-five

3.4. SPRINT PLANNING

Este proyecto se dividió en 6 Sprint cada uno con una duración de 2 semanas. en los cuales se realizan las tareas definidas en las historias de usuarios, en la siguiente Tabla 5 se mostrarán los Sprint con las historias de usuario que se realizarán.

Tabla 4. Product Backlog Detallado

Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5
H1	H3	H7	H9	H13
H2	H4	H8	H10	H14
	H5		H11	
	H6		H12	

3.5. ESTIMACIONES DE LOS SPRINTS

En las estimaciones determinamos un aproximado de cuánto tiempo va a durar cada tarea, en el siguiente Tabla 6 se mostrará cada sprint con sus tareas y el tiempo de duración de cada una.

Tabla 5. Estimaciones de los Sprints

Sprint	ID Historia de usuario	ID Tarea	Puntos de estimaciones
Sprint 1	H1	H1.1	1
		H1.2	3
		H1.2	3
		H1.4	5
	H2	H2.1	3

		H2.2	3
Sprint 2	H3	H3.1	1
		H3.2	5
		H3.3	5
	H4	H4.1	1
		H4.2	5
		H3.3	3
	H5	H5.1	5
		H5.2	2
	H6	H6.1	1
		H6.2	5
		H6.3	3
Sprint 3	H7	H7.1	5
		H7.2	3
	H8	H8.1	5
		H8.2	5
		H8.3	3
Sprint 4	H9	H9.1	8
		H9.2	3
	H10	H10.1	5
		H10.2	3
		H10.3	3
		H10.4	3
	H11	H11.1	8

		H11.2	3
	H12	H12.1	3
		H12.2	3
Sprint 5	H13	H13.1	8
	H14	H14.1	8

3.6. Sprin1

Para comenzar con este proyecto primero se configuraron diferentes herramientas y la creación de diferentes servicios para ser utilizados posteriormente. En esta primera parte se instaló un IDE para poder programar este proyecto. Se configuraron lo servicio que se van a utilizar en este proyecto. Y por último se crearon reglas de programación para ayudar a escribir código de calidad y corregir errores de sintaxis automáticamente.

Al momento de crear aplicaciones tenemos que considerar cual lenguaje de programación y que IDE (Integrated Development Environment) o entorno de desarrollo vamos a utilizar. Para este proyecto se seleccionaron 2 lenguajes de programación java script para el front-end y Python para el back-end. Como IDE se seleccionó Visual Studio Code.

Se selección Python como el lenguaje de programación del back-end por varias razones. Este es un lenguaje de programación fácil de entender, muy sencillo de utilizar y El lenguaje Python ofrece la posibilidad de usarlo en muchos dispositivos y sistemas operativos, ya que se han creado intérpretes para Unix, Linux, Windows y sistemas Mac Os. Tiene un gran soporte por la comunidad, esto es de gran utilidad debido a que si existe algún problema existen muchos foros en los cuales te pueden ayudar a resolverlo.

Se selecciono java script como el lenguaje de programación del front-end por varias razones. Debido a que en este proyecto se va a crear una aplicación móvil en los 2 sistemas operativos dominantes IOS y Android java script es el prefecto porque contiene un conjunto de herramientas, librerías y servicios los cuales te permiten desarrollar aplicaciones nativas en iOS y Android. También este es un lenguaje de programación fácil de utilizar y con una comunidad bastante activa la cual te ayuda si tienes algún problema.

Debido a que se van a utilizar 2 lenguajes se necesita un IDE que pueda soportar varios lenguajes de programación, en este caso el indicado es Visual studio code. En este también podemos instalar una herramienta llamada Prettier la cual formatea el código cada vez que guardamos un archivo. Para lograr esto tenemos que ir a extensiones e instalar Prettier y Python, no se tiene que instalar JavaScript porque está ya viene instalada por defecto en el IDE.

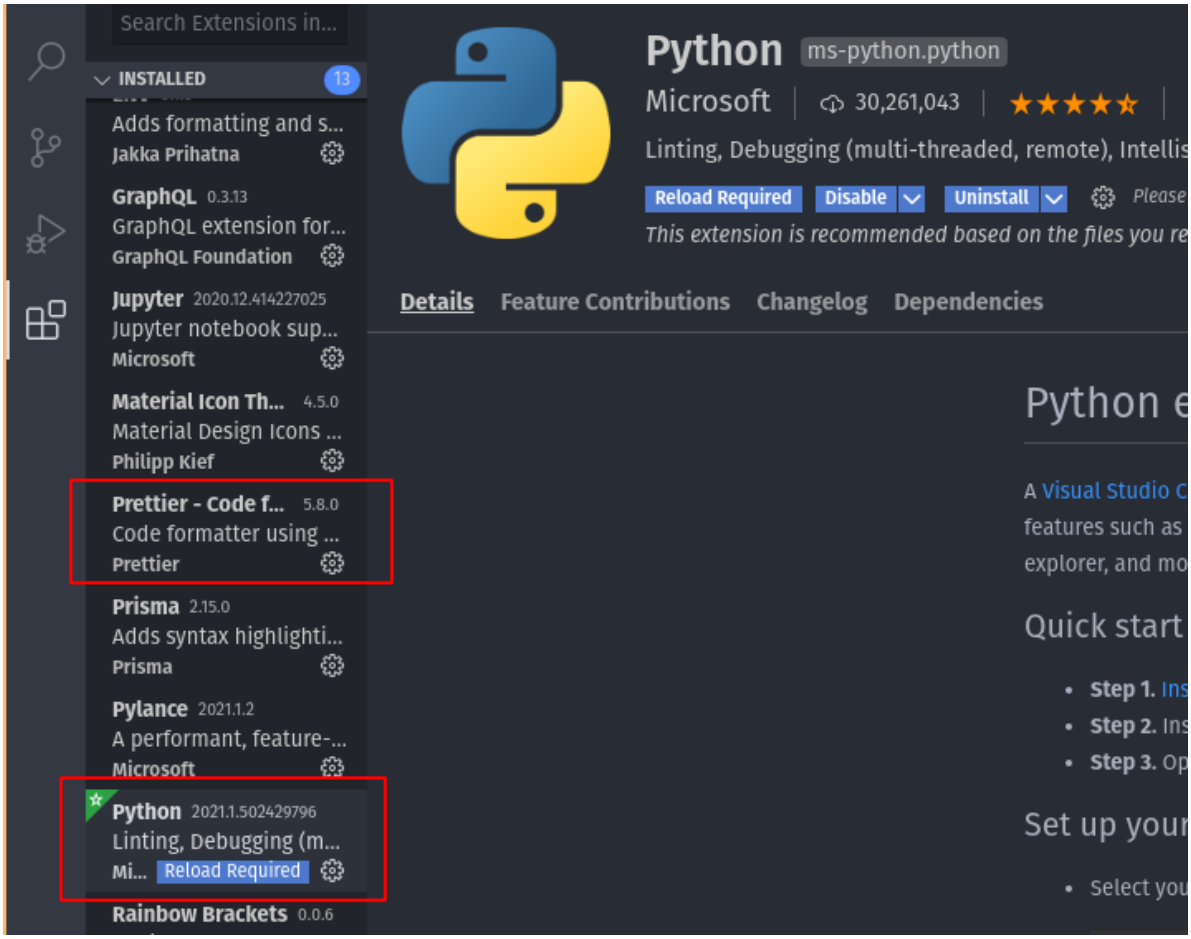

Figura 8. Configuración de herramientas de desarrollo

Como segunda tarea en este sprint configuramos AWS (Amazon Web Services), aquí se creó un usuario para poder acceder a los servicios que necesitamos, también se creó un user pool con cognito y se configuró el servidor.

Es de vital importancia crear un usuario en AWS con IAM porque este permite acceder solo a los servicios requeridos en este proyecto, de esta manera aseguramos que si por alguna razón las credenciales son expuestas no se pueda acceder a todos los servicios. En este caso Necesitaría servicios de administrador

para Lambda y Cognito. Para esto en AWS se selecciona la opción de IAM y se da acceso programático, este acceso solo permito usar AWS en línea de comandos, no permito ingresar a la consola.

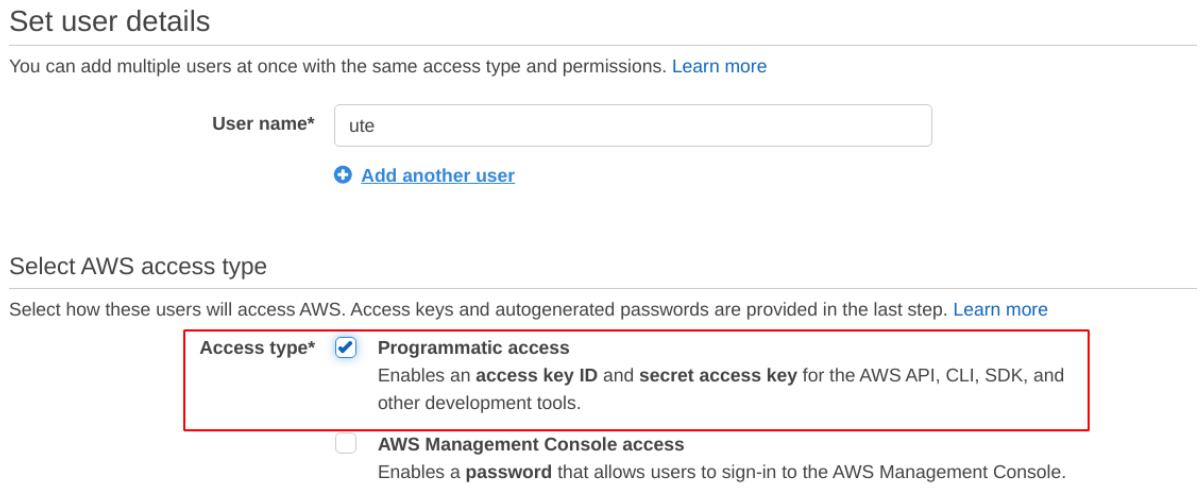

Figura 9. Creación de usuario AWS

Ahora se le da los permisos requeriidos en este proyecto. En lambda se le otorga el permiso de "AWSLambda_FullAcces" este permiso da acceso a toda la funcionalidad de lambda y para cognito se le otorga el permiso de AmazonCognitoDeveloperAuthenticatedIdentities".

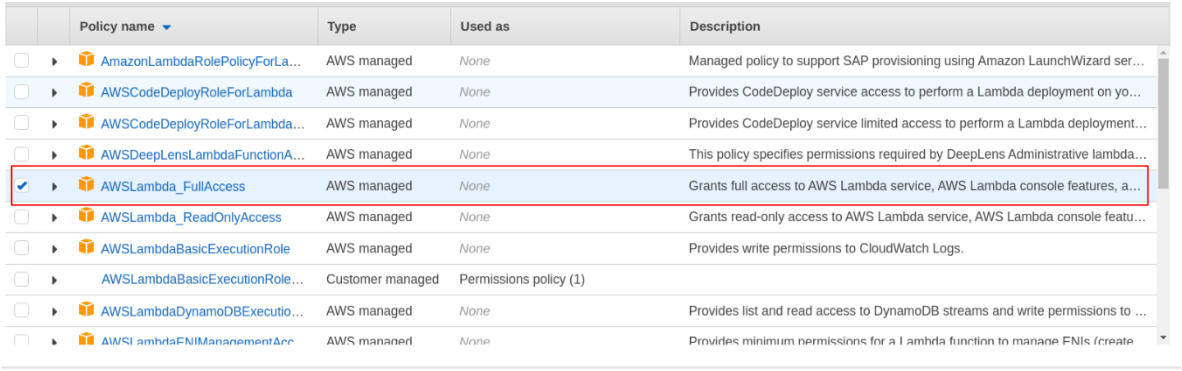

Figura 10. Permisos de usuario AWS

Se crea el usuario y se guarda las llaves en un lugar seguro, estas llaves permiten acceder a los servicios de AWS previamente configurados.

Figura 11. Llaves de acceso de cuenta AWS

Como segundo paso en esta tarea se creó un user pool en congnito, eso se utiliza para dar seguridad a la aplicación, cognito ayuda a manejar los usuarios y sus tokens para asegurar la aplicación, de esta manera solo los usuarios registrados pueden acceder a la aplicación.

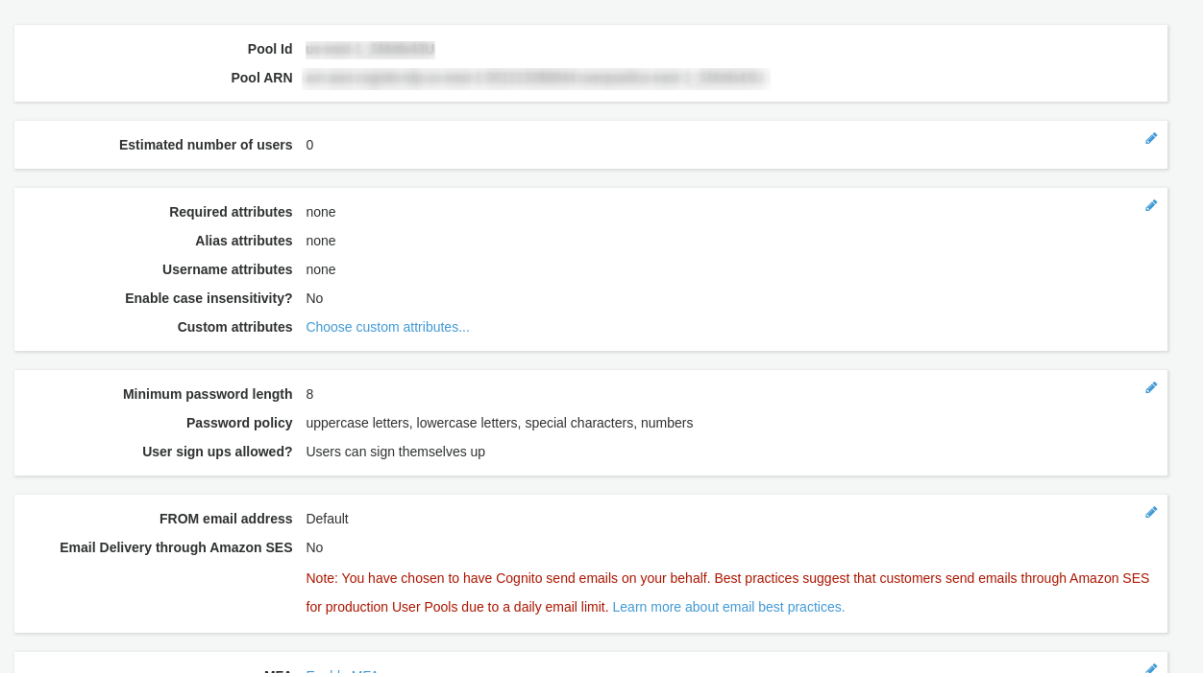

Figura 12. Configuración de cognito

Final mente en esta tarea se crea una función lambda la cual nos funciona como un servidor. Se opto utilizar una función lambda de AWS como servidor debido a que esta es mucho más barata, más fácil de controlar y de realizar mantenimiento que un API Gateway de AWS. En este caso no se configuro la función lambda desde la consola sino desde el condigo porque vamos a utilizar serverless, con serverless controlamos y hacemos el mantenimiento de la función con mucha más facilidad que hacerlo desde la consola.

```yaml
service: thot-server

provider:
  name: aws
  runtime: python3.8
  region: us-east-1
  environment:
    MONGO_URI:
    POOL_ID:
    APP_CLIENT_ID:

plugins:
  - serverless-wsgi
  - serverless-python-requirements

package:
  excludeDevDependencies: true
  exclude:
    - node_modules/**
    - venv/**
    - .serverless/**

custom:
  wsgi:
    app: src/main.app
    packRequirements: false
  pythonRequirements:
    zip: truke

functions:
  graphql:
    name: thot-server-gql
    handler: wsgi_handler.handler
    events:
      - http:
          path: graphql
          method: post
          cors: true
```

Figura 13. Configuración Serverless Python

Para esta tarea se decidió utilizar Mongo atlas debido a que Atlas maneja toda la complejidad de implementar, administrar y actualizar MongoDB. También se puede escoger un proveedor en la nube como AWS, Azure y GCP, en este caso se utilizó AWS. También Atlas tiene un cupo gratuito el cual permite utilizarlo hasta que la memoria de este llegue a su tope, esto es muy conveniente en un ambiente de desarrollo debido que aquí no se utilizan muchos recursos.

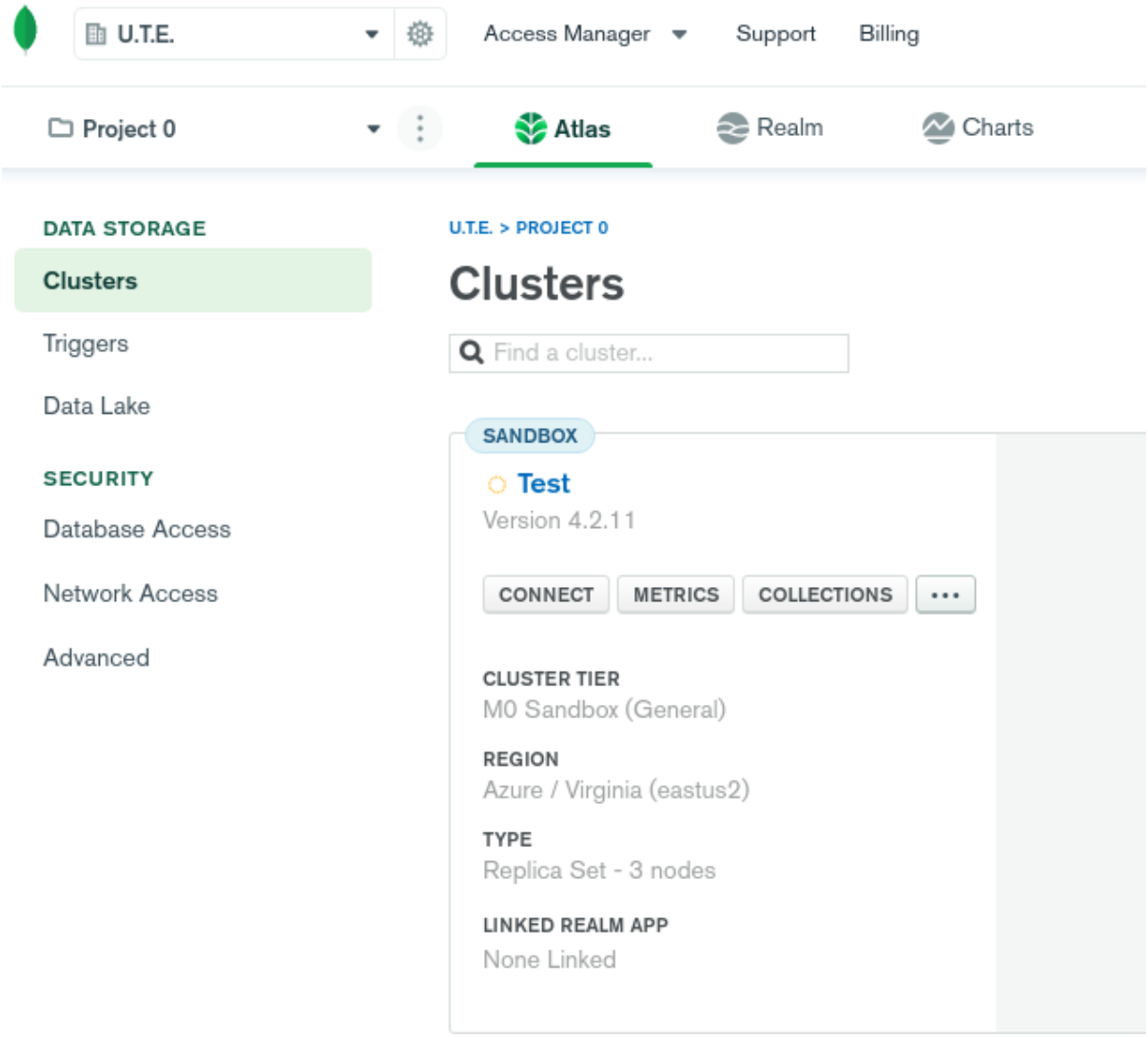

Figura 14. Creación de usuario AWS

Lint es una herramienta para identificar e informar sobre patrones encontrados en código ECMAScript / JavaScript, con el objetivo de hacer que el código sea más consistente y evitar errores. Como ya existen muchas reglas de lint en internet se utilizó una muy famosa '@react-native-community', este es un estándar que se usa bastante para crear aplicaciones en react native.

Figura 15. Configuración de link para reac native

Visual Studio code permite activar el comando para estilizar el código automáticamente cada vez que guardamos un archivo para esto se necesita las extensiones de “prettier” y “ESLint”, una vez descargadas estas extensiones se modifica los settings y se selecciona la opción de “Format on Save”.

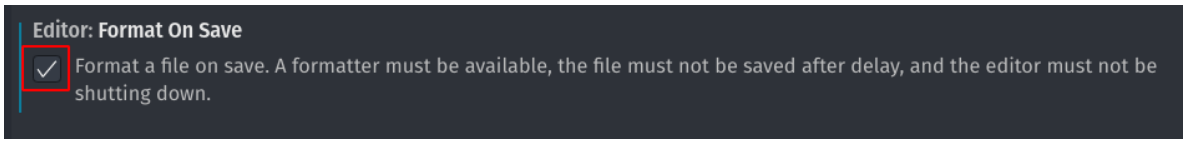

Figura 16. Configuración Visual Studio Code

3.7. Sprint 2

Para esta segunda parte del proyecto se comienza a construir el back-end, esto es de alta importancia porque el back-end se encarga de la lógica del proyecto, esta puede ser una API cuyas siglas significan “Application Programming Interfaz”. En la API de este proyecto se realizan las consultas con la base de datos y se manejan los permisos de usuario los cuales establecen si puede realizar una petición. En esta también se realiza el análisis de los datos obtenidos de los diferentes dispositivos IoT. Otra función que realiza este back-end es la interacción entre los dispositivos IoT y la base de datos mediante un microservicio MQTT. Primero se comenzará con las mutaciones las cuales nos permiten crear usuarios dispositivos IoT y los datos de estos dispositivos.

3.7.1. Mutación para crear un usuario

Con esta mutación se crea un usuario en la base de datos y en cognito para poder manejar los tokens o los accesos, primero se crea un esquema para la mutación, es esquema se lo puede ver en la figura 17, los inputs son todas las variables que se requiere para crear el usuario, después se continua con el resolver este resuelve la mutación y en caso de éxito se retornar un booleano determinando se la petición fue exitosa o no, esto está en la figura 18.

```python
class CreateUserInput(graphene.InputObjectType):
    username = graphene.String(required=True)
    password = graphene.String(required=True)
    given_name = graphene.String(required=True)
    family_name = graphene.String(required=True)
    email = graphene.String(required=True)
```

Figura 17. Input de creación de usuario

```python
class CreateUser(graphene.Mutation):
    class Arguments:
        input = CreateUserInput(required=True)

    success = graphene.Boolean()

    def mutate(self, info, input=None):
        user = Cognito(os.environ.get('POOL_ID'),
                       os.environ.get('APP_CLIENT_ID'))
        user.set_base_attributes(
            email=input.email, given_name=input.given_name, family_name=input.family_name)
        user.register(input.username, input.password)

        return CreateUser(True)
```

Figura 18. Mutación creación de usuario

3.7.2. Mutación crear dispositivo IoT

Esta mutación es muy importante para poder listar todos los dispositivos IoTs asociados a un usuario, y también para poder identificar los datos que provienen de cada dispositivo IoT. AL igual que la anterior mutación primero se crea un esquema para, un input con los datos para crear un dispositivo IoT, después se prosigue con el resolver y en caso se éxito vamos a retornar un booleano.

```python
class CreateDeviceInput(graphene.InputObjectType):
    name = graphene.String()
    device_type = graphene.String()
```

Figura 19. Input creación de dispositivo

Figura 20. Mutación creación de dispositivo IoT

```
class Create_Device(graphene.Mutation):
    class Arguments:
        input = CreateDeviceInput(required=True)

    device = graphene.Field(Device)

    def mutate(self, info, input=None):
        user = get_user(info.context)

        id = db.insert({
            'name': input.name,
            'device_type': input.device_type,
            'owner_email': user.email,
            'created_at': datetime.now(),
            'updated_at': datetime.now()
        })

        device = db.find_one({'_id': ObjectId(id)})

        myDevice = Device(
            id=str(ObjectId(device['_id'])),
            name=device['name'],
            device_type=device['device_type'],
            owner_email=device['owner_email'],
            created_at=device['created_at'],
            updated_at=device['updated_at']
        )

        return Create_Device(myDevice)
```

3.7.3. Mutación para editar el dispositivo IoT

Esta mutación sirve para editar el dispositivo IoT, esta mutación tiene como propósito cambiar el nombre y el tipo de dispositivo, el objetivo de esta es que cuando el usuario se equivocó al momento de crear un dispositivo este lo pueda cambiar.

3.7.4. Mutación para crear un evento de un dispositivo

Esta mutación alimenta la base de datos con datos provenientes de los dispositivos IoT, estos datos son los que serán analizados. El análisis de los datos se lo realizara en el siguiente sprint, por el momento solo se crean los eventos.

3.8. Sprint 3

Para esta tercera parte del proyecto se listarán los dispositivos IoT por usuario y se creara una suscripción en la cual se analizarán y se mostraran en pantalla el estado actual del dispositivo IoT, a su vez también se muestran los datos ya procesados en tiempo real.

3.8.1. Query para mostrar los dispositivos IoT

Esta query tiene como objetivo mostrar todos los dispositivos IoT que tiene el usuario disponible, también sirve para que el usuario pueda saber de cual dispositivo IoT vienen los datos. Para el resolver de esta query se va a retornar un arreglo con todos los dispositivos de un usuario.

```python
def resolve_page_of_devices(root, info, pageRequest):
    token = check_token(info.context)

    if token == True:
        raise GraphQLError('Token is expired')

    devices = []

    if pageRequest.cursor is None:
        cursor = db.find().limit(pageRequest.limit)
    else:
        cursor = db.find(
            {'_id': {'$gt': ObjectId(pageRequest.cursor)}}
        ).limit(pageRequest.limit)

    for doc in cursor:
        devices.append(Device(
            id=str(doc['_id']),
            name=doc['name'],
            device_type=doc['device_type'],
            owner_email=doc['owner_email'],
            created_at=doc['created_at'],
            updated_at=doc['updated_at']
        ))

    if not devices:
        return PageOfDevices(devices=[], has_next_page=False, last_cursor='NULL')

    has_next_page = not cursor.count() == cursor.count(with_limit_and_skip=True)
    last_cursor = devices[-1].id

    return PageOfDevices(devices=devices, has_next_page=has_next_page, last_cursor=last_cursor)
```

Figura 21. Query para mostrar dispositivos IoT

3.8.2. Suscripción para el detalle de un dispositivo

Para este paso primero se creará diversas funciones para poder obtener datos estadísticos los cuales ayuden al usuario a comprender lo que está sucediendo con el dispositivo IoT. Los datos principales son una tabla de frecuencia con la cual un dato se repite en un intervalo de tiempo, otro dato muy importante es la distribución normal la cual ayuda a comprender como los datos están estructurados.

```
def resolve_frequency_table(root, info, device_id):
    cursor = events.find({'device_id': device_id})
    data = json.loads(json_util.dumps(cursor))

    try:
        df = pd.json_normalize(data, 'samples', ['day'])

        values, count_freq = np.unique(df['value'], return_counts=True)
        frequency_table = []

        for table in np.asarray((values, count_freq)).T:
            frequency_table.append(Table(
                x=str(table[0]),
                y=table[1]
            ))

        return frequency_table

    except:
        return[]
```

Figura 22. Suscripción para mostrar información en tiempo real

3.9. Sprint 4

En este sprint se comienza con la parte grafica de la aplicación, para esto se decidió utilizar React Native porque esta permite utilizarla en los 2 sistemas operativos móviles más utilizados del mundo Android y IOS. Android es un sistema operativo móvil desarrollado por Google y actualmente es el más utilizado debido a que es basado en el kernel de Linux y otros softwares de código abierto. IOS es un sistema operativo móvil desarrollado por Apple el cual solo puede ser utilizado en los dispositivos que cuenten con esta marca, en la siguiente figura se mostrará un gráfico con los sistemas operativos móviles más utilizados.

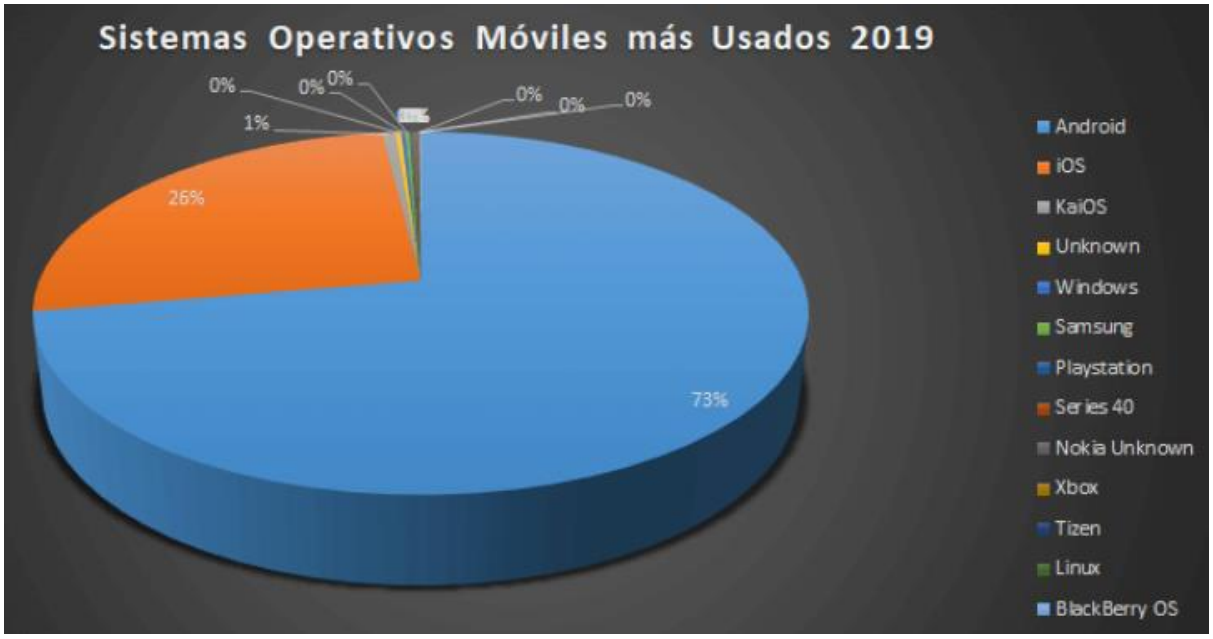

Figura 23: Sistemas Operativos Móviles más usados (Ramírez, 2019).

3.9.1. Lógica de la aplicación

Para esta primera parte se define la lógica de la aplicación, se configura lo que el ambiente y se crean diversos componentes. Primero vamos a crear varios componentes para reutilizarlos, esto nos ayuda a que el código no sea repetitivo y poder reutilizar componentes en todo nuestro código. También se configurará cognito para el lognin de la aplicación.

Primero creamos los componentes para los inputs, cards, títulos, textos, mensajes, botones, menús y gráficos estadísticos. Al crear estos componentes podremos reutilizarlos cuando queramos y tendríamos un estándar de lo que vamos a utilizar.

```
const FrequencyChart = ({ deviceId, type }) => {
  const { data, loading } = useQuery(GET_FREQUENCY, {
    variables: { deviceId },
  });

  if (loading) {
    return <Loading />;
  }

  return (
    <VictoryChart domainPadding={10}>
      <VictoryBar data={data.frequencyTable} />
      <VictoryAxis
        label={type}
        style={{
          axisLabel: { padding: 30 },
          tickLabels: { angle: 45 },
        }}
      />
      <VictoryAxis dependentAxis />
    </VictoryChart>
  );
};
```

Figura 24: Componente para mostrar gráficos.

Después vamos a configurar cognito, esto nos ayuda a crear un login mucho más fácil y rápido, solo tenemos que instalar la dependencia de “aws amplify”

```
Amplify.configure({
  identityPoolId: '[redacted]',
  region: '[redacted]',
  identityPoolRegion: '[redacted]',
  userPoolId: '[redacted]',
  userPoolWebClientId: '[redacted]',
});
```

Figura 25. Configuración de Cognito en React Native.

3.9.2. Visualización de los dispositivos IoT

Esta tarea es importante debido a que podremos mostrar los dispositivos IoT que tenemos disponibles. Primero vamos a crear una interfaz gráfica para mostrar los dispositivos. Segundo vamos a realizar una query para obtener todos los dispositivos IoT.

Para crear esta interfaz y mostrar todos los dispositivos vamos a utilizar los componentes previamente creados. Crearemos una lista con que contenga un texto con el nombre del dispositivo y el tipo de dispositivo que es.

```
return (
  <ListItem thumbnail onPress={() => history.push(`/device/${id}`)}>
    <Body>
      <Text>{name}</Text>
      <Text note numberOfLines={1}>
        {type}
      </Text>
    </Body>
    <Right>
      <Icon name='arrow-forward' />
    </Right>
  </ListItem>
);
```

Figura 26. Componente Lista de los dispositivos IoT.

Como esta API esta echa con GQL se necesita una herramienta para obtener los datos con mayor facilidad, para esto existen varias herramientas las cuales podemos utilizar, en este caso se va a utilizar Apollo Client para React.

```
export const GET_DEVICES = gql`
  query devices($pageRequest: PageRequest) {
    pageOfDevices(pageRequest: $pageRequest) {
      devices {
        id
        deviceType
        name
      }
      hasNextPage
      lastCursor
    }
  }
`;
```

Figura 27. Request al servidor de Python.

3.9.3. Detalle del dispositivo IoT

Esta tarea es importante debido a que se podrá mostrar los cálculos del dispositivo IoT. Primero se crea una interfaz gráfica para mostrar todos los datos estadísticos. Segundo se realizará una suscripción para obtener todos los dispositivos IoT.

Para crear esta interfaz y mostrar las gráficas se utilizará los componentes previamente creados y el paquete `victory-native` el cual nos facilitará el trabajo. Se crearon varios gráficos para mostrarlos, estos son actualizados en tiempo real.

Figura 28. Componente para mostrar detalle del dispositivo IoT.

```
return (
  <VictoryChart domainPadding={10}>
    <VictoryBar data={data.frequencyTable} />
    <VictoryAxis
      label={type}
      style={{
        axisLabel: { padding: 30 },
        tickLabels: { angle: 45 },
      }}
    />
    <VictoryAxis dependentAxis />
  </VictoryChart>
);
```

3.9.4. Cerrar sesión

Para esta tarea vamos a usar cognito para cerrar la sesión del usuario y dirigirse a la pantalla de inicio de la aplicación. En el siguiente código podemos observar el cerrado de cesión con cognito.

Figura 29. Cerrar la sesión de cognito.

```
const logout = async () => {
  try {
    await Auth.signOut();
    setToken(null);
    localStorage.removeItem('token');
    localStorage.removeItem('email');
  } catch (error) {
    console.log('error signing out: ', error);
  }
};
```

3.10. Sprint 5

En este sprint se construirá y se programará el dispositivo IoT. Para la construcción de este se utilizará un Arduino uno. Para la programación de este se utilizó johnny-five que pueda enviar los datos generados al servidor y este los almacene en la base de datos.

3.10.1. Construcción del dispositivo IoT

En esta tarea se creó un dispositivo IoT para que nos de información de la temperatura del ambiente. En este ejemplo vamos a monitorizar la temperatura en el proceso de fermentación de un vino, en este caso mantener un temperatura adecuado es de vital importancia. Primero realizamos un esquema de conexiones con el programa `Fritzing`. Segundo con este esquema podremos construir el dispositivo, para esto vamos a necesitar un Arduino uno el cual funciona como una computadora de un solo proceso, También un ethernet Shield para tener una

conexión a internet, un sensor DHT22 este nos dará el valor de la temperatura, una resistencia de 10k para que la electricidad no queme los circuitos y cable de conexión para conectar todo.

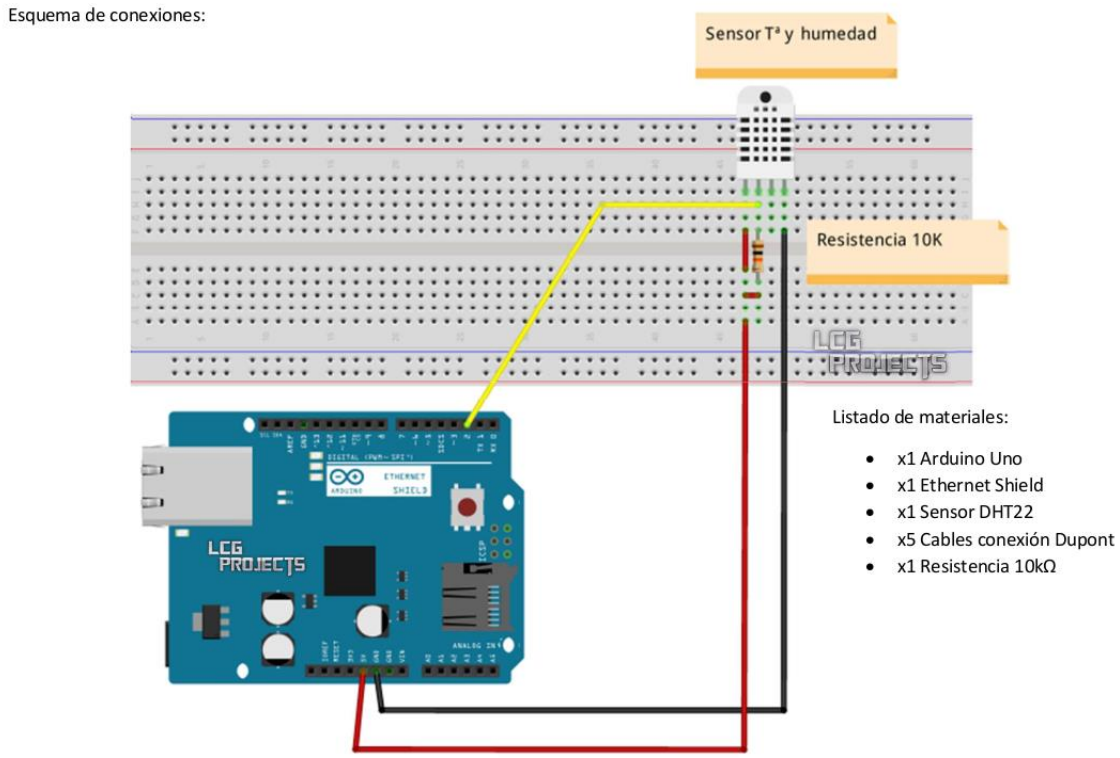

Figura 30. Plano del dispositivo IoT.

3.10.2 Programación del dispositivo IoT

Para esto se utilizó johnny-five JS este permite utilizar el lenguaje de programación JavaScript en un Arduino uno, he decidido utilizar esto porque es el lenguaje de programación con el que estoy más familiarizado y es mucho más fácil de usar.

```
board.on('ready', () => {
  const thermometer = new Thermometer({
    controller: 'DHT22_I2C_NANO_BACKPACK',
  });

  thermometer.on('change', () => {
    const { celsius } = thermometer;
    console.log('Thermometer');
    console.log('  celsius      : ', celsius);
    console.log('--------------------------------------');
    myEvent({
      variables: { input: { deviceID: process.env.deviceID, value: celsius } },
    });
  });
});
```

Figura 31. Código del dispositivo IoT.

4. CONCLUSIONES Y RECOMENDACIONES

4. CONCLUSIONES Y RECOMENDACIONES

A fin de concluir el presente proyecto a continuación se menciona las conclusiones obtenidas y recomendaciones sugeridas:

CONCLUSIONES:

- El procesamiento y análisis de los datos por parte del servidor resultaron satisfactorios para el volumen de datos en este proyecto.
- La aplicación móvil respondió bien a las solicitudes al servidor en los emuladores de Android y IOS, los datos se muestran con claridad y las gráficas ayudan en la comprensión de estos. En dispositivos movilices la respuesta en Android fue un poco más lenta que en IOS, pero esto puede ser debido a que el dispositivo Android en el que se probo es un Samsung S3 un teléfono celular del 2012 y en el dispositivo IOS en el que se probó fue un IPhone 12.
- Los servicios en la nube que se utilizaron para este proyecto tuvieron una respuesta satisfactoria el 100% de las veces, lo cual facilito bastante las pruebas y elimino los inconvenientes que se pueden dar al tener un servidor propio.
- El servidor el cual fue desarrollado en Python procesó y analizo lo datos enviados por los dispositivos con una velocidad muy aceptable, en promedio fue alrededor de 2 segundo por cada análisis de dispositivo IoT, hay que tomar en cuenta que en promedio un dispositivo IoT genera 17280 eventos en el transcurso de 24 horas.
- La construcción del dispositivo IoT fue un éxito gracias a que se realizó un diagrama de este antes de la construcción. Sin embargo, en la programación se tuvo algunos problemas al conectarse con el servidor, en un principio se utilizó Arduino IDE, pero por este problema se cambió a Johnny-Five

RECOMENDACIONES:

- Este proyecto da las bases para poder realizar un análisis de predicción en base a los datos obtenidos.
- Este proyecto se lo realizó con un bajo presupuesto, por esta razón solo se pudo probar en 2 dispositivos físicos, para un futuro lo mejor sería probar con al menos 5 dispositivos móviles.
- Para este código no se realizaron pruebas a este por falta de tiempo y porque solo una persona realizo el código. Sin embargo, para un futuro sería muy

bueno realizar pruebas en el código para obtener un programa con mucha más calidad.

BIBLIOGRAFÍA

Ágiles, M. (Pontificia Universidad Católica del Perú). *SCRUM: METODOLOGÍAS ÁGILES.* Lima, Perú: 2018.

Alley, G. (2 de 20 de 2018). *What is Data Streamin?* Obtenido de What is Data Streamin?: https://www.alooma.com/blog/what-is-data-streaming

Amazon Web Services (AWS). (2 de 10 de 2020). *¿Qué son los datos de streaming?* Obtenido de ¿Qué son los datos de streaming?: https://aws.amazon.com/es/streaming-data/

Amazon Web Services. (04 de 12 de 2019). *¿Qué son los datos de streaming?* Obtenido de ¿Qué son los datos de streaming?: https://aws.amazon.com/es/streaming-data/

Andrés Navarro Cadavid, Juan Daniel Fernández Martínez, Jonathan Morales Vélez. (2013). Revisión de metodologías ágiles para el desarrollo de software. *Reseaech Gate*, 30-39.

Dimes, T. (2015). *Conceptos Básicos de Scrum: Desarrollo de software Agile y manejo de proyectos Agile.* México: Prentice Hall.

Frédéric Combaneyre, S. B. (2017). *Understanding Data Streams in IoT.* EEUU: Springer.

Genbeta. (3 de 02 de 2014). *MongoDB: qué es, cómo funciona y cuándo podemos usarlo*. Obtenido de MongoDB: qué es, cómo funciona y cuándo podemos usarlo: https://www.genbeta.com/desarrollo/mongodb-que-es-como-funciona-y-cuando-podemos-usarlo-o-no

IBM Developer. (04 de 10 de 2017). *Conozca MQTT*. Obtenido de Conozca MQTT: https://developer.ibm.com/es/articles/iot-mqtt-why-good-for-iot/

Ifgeekthen. (8 de 04 de 2020). *Qué es NodeJS y primeros pasos*. Obtenido de Qué es NodeJS y primeros pasos: https://ifgeekthen.everis.com/es/que-es-node-js-y-primeros-pasos

Liñán Colina, A. V. (2016). *Internet of Things IN 5 DAYS. 227.*

Mansaf, A., Shakil, K. A., & Samiya, K. (2019). *Internet of Things (IoT).* EEUU: Springer.

Marc, P. D. (2019). *Seguridad en dispositivos móviles.* España: UOC Universidad Oberta de Calalunya.

Marcelo Alcaraz. (07 de 04 de 2020). *Internet de las Cosas.* Paraguay: Universidad Católica Nuestra Señora de la Asunción. Obtenido de Stream Processing with IoT Data: Challenges, Best Practices, and Techniques: https://www.confluent.io/blog/stream-processing-iot-data-best-practices-and-techniques/

Mongo DB. (12 de 12 de 2020). *MongoDB Atlas*. Obtenido de MongoDB Atlas: https://www.mongodb.com/cloud/atlas

Node JS. (15 de 01 de 2018). *Node JS*. Obtenido de Node JS: https://nodejs.dev/

Pereira Villazón, T. P. (2019). Big data y Relaciones Públicas. Una revisión bibliográfica del estado de la cuestión. *Comunicación*, 18-22.

Red Hat. (12 de 12 de 2020). *¿Qué es el Internet de las cosas?* Obtenido de ¿Qué es el Internet de las cosas?: https://www.redhat.com/es/topics/internet-of-things/what-is-iot

Red Hat. (15 de 09 de 2020). *Big Data*. Obtenido de Big Data: https://www.redhat.com/es/topics/big-data

Ticportal. (12 de 10 de 2019). *Amazon Web Services*. Obtenido de Amazon Web Services: https://www.ticportal.es/temas/cloud-computing/amazon-web-services

Universidad de Sevilla. (12 de 10 de 2020). *¿Para qué sirve Python?* Obtenido de ¿Para qué sirve Python?: https://www.esic.edu/rethink/tecnologia/para-que-sirve-python#:~:text=El%20lenguaje%20de%20programaci%C3%B3n%20Python,aplicaciones%20empresariales%20fiables%20y%20escalables.

Wikipedia. (06 de 01 de 2021). *Amazon Web Services*. Obtenido de Amazon Web Services: https://es.wikipedia.org/wiki/Amazon_Web_Services