# **IMPLEMENTACIÓN DE SERVIDOR DE GESTIÓN WEB PARA LA MONITORIZACIÓN DEL ESPECTRO DE LAS BANDAS COMERCIALES DE TVD**

**Isaac Vega Rodríguez, Rufino Cabrera Alvarez, Dreyelian Morejón Betancourt, Ariel Catalá Valencia,** **Ana Ivis Mena Valdés**

Universidad Tecnológica de La Habana José Antonio Echeverría (**CUJAE**),***LACETEL***, Instituto de Investigación y Desarrollo de Telecomunicaciones, La Habana, Cuba

e-mail: [isaac@est.tele.cuaje.edu.cu](mailto:isaac@est.tele.cuaje.edu.cu)

e-mail: [rufino@lacetel.cu](mailto:rufino@lacetel.cu)

e-mail: [dreyelian@lacetel.cu](mailto:dreyelian@lacetel.cu)

e-mail: [ariel@est.tele.cuaje.edu.cu](mailto:ariel@est.tele.cuaje.edu.cu)

e-mail: [amena@lacetel.cu](mailto:amena@lacetel.cu)

RESUMEN

Actualmente en ***LACETEL***®, Instituto de Investigación y Desarrollo de Telecomunicaciones, está siendo desarrollada una solución para la monitorización del espectro radioeléctrico en las bandas comerciales de televisión. Para ello se prevé el despliegue de un conjunto de sensores distribuidos geográficamente, los cuales realizaran barridos periódicos del espectro, medirán los parámetros fundamentales de la señal detectada y lo enviaran a un nodo central para su gestión y procesamiento. El presente trabajo se enfoca en el diseño web del sistema de gestión que realiza el procesamiento de los datos. La propuesta está realizada haciendo uso de los lenguajes HTML5, CSS, JavaScript y TypeScript. El presente documento pretende explicar las diferentes secciones que componen el proyecto.

PALABRAS CLAVES: Monitorización del espectro radioeléctrico, Sistema de gestión de datos, Diseño web, Televisión Digital.

# WEB MANAGEMENT SERVER IMPLEMENTATION for SPECTRUM MONITORING of DTV COMERtial band

ABSTRACT

Currently in LACETEL®, Research and Development Telecommunications Institute, has been developed a solution for monitoring the radio spectrum in commercial television bands. To achieve this goal, the deployment of a set of geographically distributed sensors is planned, in order to perform periodic sweeps of the spectrum, measuring the fundamental parameters of the detected signal and sending them to a central node for its management and processing. The current work focuses in the web design of the management system, which it processes the data information. The proposal was made using HTML5, CSS, JavaScript and TypeScript. The current document pretends to explain the different sections of the project.

KEY WORDS: Monitoring of the radio spectrum, Data management system, Web design, Digital Television.

Introducción

Como mencionan los autores en [[1](#_ENREF_1)], con el despliegue de la televisión digital (TVD) en el país, surge también la necesidad de monitorizar en tiempo real la calidad de la señal puesta en el aire por cada uno de los centros transmisores. No obstante, la ubicación del equipamiento necesario para este propósito constituye un reto, tanto desde el punto de vista logístico como económico, debido a la gran cantidad de emplazamientos involucrados en el proceso. Contar con un sistema de monitorización para el seguimiento de parámetros seleccionados de la señal y su registro, permitiría garantizar un mejor servicio partiendo de la observación y análisis de los parámetros monitorizados. Este escenario propició la búsqueda de soluciones más económicas, siendo un caso de estudio el empleo de receptores comerciales de televisión digital. [[1-4](#_ENREF_1)]

La no disponibilidad de un sistema para el seguimiento y adquisición de información sobre el comportamiento de la señal DTMB con posibilidad de ser accedido y gestionado remotamente, ha llevado al estudio de este tipo de soluciones. Los sistemas de monitorización y control son capaces de obtener información del entorno donde se despliegan para su posterior análisis y realizar las acciones pertinentes. Para obtener la información de su entorno hacen uso de una red de sensores desplegada en puntos específicos a conveniencia de los especialistas. Además, permiten hacer un seguimiento de los valores recopilados por todos los sensores que forman la red. Estos sistemas de monitorización dispondrían de una interfaz para la visualización de los datos capturados en la pantalla de nuestro ordenador, teléfono móvil o PDA. También se podrían obtener estadísticas, gráficas, realizar consultas a un historial de datos, entre otras acciones. [[5-9](#_ENREF_5)]

Actualmente en ***LACETEL***®, Instituto de Investigación y Desarrollo de Telecomunicaciones, está siendo desarrollada una solución para la monitorización del espectro radioeléctrico en las bandas comerciales de televisión. Para ello se prevé el despliegue de un conjunto de sensores distribuidos geográficamente, los cuales realizarán barridos periódicos del espectro, medirán los parámetros fundamentales de la señal detectada y lo enviarán a un nodo central para su gestión y procesamiento. El empleo de equipos de precisión para esta tarea, representa una limitación dado su alto costo y la necesidad de efectuar mediciones en un número elevado de ubicaciones de difícil acceso [[10-12](#_ENREF_10)]. Adicionalmente, resultaría ventajosa la inclusión de funcionalidades de conectividad en red, aspecto que permitiría el aprovechamiento de las infraestructuras de comunicación existentes y proveerían acceso universal al sistema desde cualquier punto de la geografía nacional. En este sentido, el paradigma de Internet de las Cosas (IoT) ofrece un modelo conveniente sobre el cual desarrollar el sistema, ofreciendo la flexibilidad, escalabilidad y funcionalidad necesarias para dar solución al problema planteado [[5-9](#_ENREF_5), [13-16](#_ENREF_13)]. [[1-4](#_ENREF_1)]

Tres de los parámetros que fundamentales para caracterizar el comportamiento de la señal de radio frecuencia son la intensidad o potencia recibida en el extremo receptor, la relación señal al ruido y la razón de bit erróneo. Por ello, la obtención de un prototipo funcional de sistema de monitorización de la señal DTMB con soporte de conectividad en red para el almacenamiento y gestión de estos tres parámetros ha constituido la meta principal de este proyecto. Para dar cumplimiento al mismo se definieron tres líneas principales de trabajo, el desarrollo de la Plataforma Software para la recolección y envío de las mediciones de los parámetros de la señal [1-4], el de la Plataforma Hardware con soporte de conectividad en red [4] y la Plataforma Software para la recolección y envío de las mediciones de la señal, siendo esta última es el punto central del presente trabajo. Este documento se enfoca en la descripción del diseño del sistema de gestión de esta información a partir del diseño web que realice dicho procesamiento y visualización de la información.

Punto de partida y estado del arte

Actualmente no se cuenta con un desarrollo inicial, por lo que se parte de cero seleccionando los lenguajes de implementación. Los sistemas de información actuales se basan en bases de datos (BD) y sistemas de bases de datos (SGBD) que se han convertido en elementos imprescindibles de la vida cotidiana de la sociedad moderna. Este documento describe la selección de la base de datos a utilizar y los elementos esenciales que condujeron a la implementación de la web que gestione y visualice la información recibida por los sensores de los parámetros de la señal DTMB.

Los tres parámetros (potencia de la señal, relación señal al ruido y razón de bit erróneo) seleccionados deben ser visualizados en forma de gráficos, mapas y eventos. Los gráficos representarán uno de los parámetros previamente seleccionado por el usuario y mostrarán el comportamiento de la recepción de uno o varios sensores. Los mapas representarán geográficamente la ubicación de los transmisores y sensores e identificarán mediante una escala de colores el rango de valores que recibe cada sensor. Los eventos mostrarán un grupo de estadísticas según la interpretación de la señal como, por ejemplo, la caída drástica de un parámetro determinado de la señal, alertas de no recepción de la misma, etc. Además, las estadísticas deben almacenarse en el tiempo, de acuerdo al tamaño en disco, de forma que los valores históricos puedan ser analizados para realizar futuras predicciones del comportamiento de la señal.

En los últimos años se han desarrollado una amplia variedad de herramientas que permiten analizar series temporales, ya sea para analizar la subida o caída de los precios en el mercado, o las fluctuaciones en el precio de una criptomoneda, el producto interno bruto, la temperatura ambiental en diferentes puntos, parámetros relativos a la salud, en fin, se puede analizar cualquier dato que pueda ser referenciado a un momento específico. Realizar análisis sobre estos datos, no solo nos da una idea de cómo se han comportado a través del tiempo, sino de cuál puede ser su comportamiento a medida avanza el tiempo y así poder tomar un camino u otro en dependencia de la predicción realizada. Entre las muchas herramientas que existen para efectuar dicha visualización y análisis están: Grafana, netdata, Prometheus, Kibana, Datadog, Graphite entre otros.

Bases de datos

Existen muchas herramientas que permiten gestionar la base de datos, como es el caso de MariaDB, MongoDB, CouchDB, InfluxDB entre muchas otras. Cada una tiene sus ventajas y desventajas según criterios de comparación como son la rapidez, flexibilidad, eficiencia de gestión para grandes volúmenes de datos, capacidad de respuesta ante una gran concurrencia de consultas entre otros aspectos. Para este proyecto, debido a que el mayor peso de la base de datos recae en la manipulación de las señales provenientes de los sensores para su posterior análisis, se procedió a seleccionar InfluxDB [[17](#_ENREF_17)] por las siguientes características:

* Es de código abierto, por lo que no es necesario pagar ningún tipo de licencia para su uso.
* Está optimizado para guardar y manipular series temporales.
* Soporta una gran concurrencia de consultas.
* Permite obtener el resultado de la consulta, en diferentes formatos (tabla, JSON etc....).
* El equipo de desarrollo de InfluxDB también mantiene un conjunto de herramientas especializadas en el manejo de series temporales, como es el caso del stack TICK (Telegraf, InfluxDB, Cronograf y Kapacitor).
* La base de datos ocupa poco espacio en disco en comparación con otras bases de datos (MongoDB, MariaDB, etc....).

Selección del servidor

En el caso del servidor web, que es el encargado de proveer de los servicios necesarios para que todo el sistema de monitoreo funcione de manera fluida, también se exploraron múltiples opciones, entre las que se encuentran Nginx, Apache, ASP.NET, NodeJS etc... Se seleccionó NodeJS [[18](#_ENREF_18)] como herramienta para implementar el servidor, debido a que:

* Es de código abierto.
* Puede manejar una gran cantidad de peticiones de forma concurrente con un tiempo de respuesta muy pequeño.
* Utiliza el V8, que es el motor de interpretado de JavaScript, creado y mantenido por Google.
* Posee una infinidad de módulos que se pueden utilizar para hacer casi cualquier cosa, desde un servidor sencillo hasta una aplicación de escritorio.
* Se programa en JavaScript, que es el mismo lenguaje que se utiliza en el lado del cliente, por tanto, existe una total compatibilidad entre la página en sí, y el servidor en cuanto a lenguaje.
* Tiene como apoyo una comunidad de desarrolladores inmensa, lo que implica que siempre está en constante desarrollo, añadiendo cada vez nuevas y más significativas mejoras tanto en el rendimiento como en el uso de las bondades que brinda.

Conformación del cliente web

Dentro del servicio de monitoreo, se cuenta también con una interfaz web que permite interactuar con dicho servicio en cuestión. Hoy en día existen muchísimas librerías y plataformas para desarrollar páginas o servicios web de forma rápida, sencilla y poco propensa a errores. Como parte de las opciones se encuentran: jQuery, AngularJS, Angular, React, Vue, entre otros. Para este proyecto se seleccionó Angular [[19](#_ENREF_19)] como plataforma de desarrollo por las siguientes razones:

* A diferencia de la programación web tradicional en JavaScript, el desarrollo en Angular se realiza en TypeScript, que es un superconjunto de JavaScript, pero tiene como ventaja el uso de tipos de variables, clases, enumeraciones, y todo un conjunto de bondades que hacen que el desarrollo de la página sea conciso y enfocado en buenas prácticas de desarrollo.
* Su filosofía está orientada al uso de componentes, de forma tal que cada parte de la página se puede encapsular en un componente y modelar cada uno de forma distinta y separada sin que se interfieran las funciones de un componente con otro. Entiéndase por componente, a una lista, menús, formularios, etc...
* La página que se genera es compatible con la mayoría de los navegadores gracias a que antes de exportar el proyecto, Angular detecta cuáles son las funcionalidades que se han agregado nuevas al lenguaje en los últimos años, y las reimplementa para asegurar dicha compatibilidad.
* Se pueden manejar fácilmente aspectos como la autenticación de usuarios, el acceso a las diferentes rutas de la página, protegiendo las mismas ante usuarios no autorizados.
* Permite crear aplicaciones web con un alto rendimiento y facilidad de mejorar, mantener y probar las mismas.

Una vez seleccionados los componentes esenciales para el desarrollo del sistema de monitorización, se analizan entonces las necesidades y objetivos a plantearse con el fin de tener bien enfocado y encaminado el proyecto.

**DESARROLLO**

Estructura general del proyecto

Existen 3 partes fundamentales del proyecto: la base de datos, el servidor y el cliente web. La Figura 1 muestra cada una de las partes. Nótese que cada elemento, utiliza como fuente de datos el elemento con el cual está relacionado por la izquierda. Aunque forman parte del proyecto, solo se abordarán: la base de datos, el servidor y el cliente web.

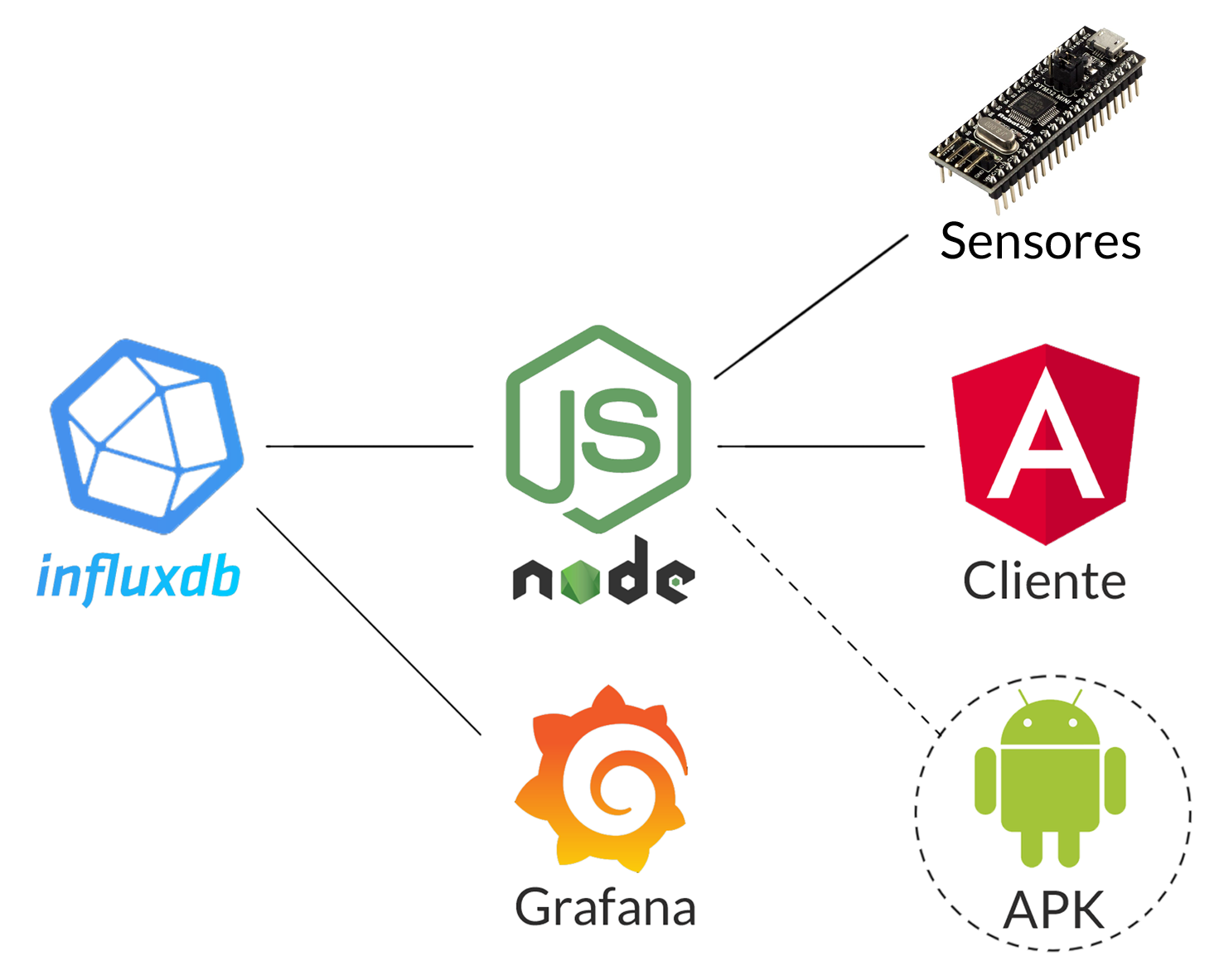


Figura 1. Diagrama de dependencias del proyecto.

Dado que el servidor tiene implementada una API (Interfaz de Programación de Aplicaciones), se puede utilizar tanto desde un cliente web como desde una aplicación móvil. Aunque esta última no está dentro del enfoque del presente trabajo, se puede considerar una opción para planes futuros.

Base de datos

Una base de datos es un conjunto de datos pertenecientes a un mismo contexto, los cuales son almacenados de forma sistemática para su posterior uso. La base de datos seleccionada fue InfluxDB, la cual es una base de datos relacional, o sea, guarda sus datos en forma de tablas o relaciones. Cada tabla que se guarda dentro de la base de datos, posee un nombre único, el cual identifica el dato en cuestión que se desea guardar, por ejemplo “Señales”, “Usuarios” y así sucesivamente. Cada una de las tablas, tiene una cantidad de columnas fijas, asociadas a las características de cada uno de los elementos de dicha tabla, como podría ser, dentro de la tabla “Usuarios”, el “nombre”, “edad”, “email” etc...

Las tablas con las que debe contar dicho sistema son las siguientes:

* Usuarios: El contenido de esta tabla, debe estar orientado al manejo de los usuarios del sistema. De cada usuario se deben guardar los datos personales esenciales, así como otros datos relevantes como el nivel de privilegios (administrador, moderador, usuario etc....).
* Sensores: De cada sensor que tendrá acceso a guardar en la base de datos, se debe guardar su identificador, así como sus coordenadas geográficas para ser ubicados en el mapa.
* Señales: En esta tabla se deben guardar las muestras obtenidas por los sensores, así como la fecha y hora exacta en que se envió, así como el identificador del sensor desde el cual se leyeron dichos parámetros.
* Otras tablas como: Provincia, Servicio, Transmisor etc.... donde cada una contiene los datos pertinentes.

Por defecto, el gestor de base de datos de InfluxDB despliega un servidor HTTP con el cual se puede interactuar directamente, aunque en el proyecto no se utilizará de forma directa sino a través de un módulo de NodeJS llamado **influxdb-nodejs**. Este gestor se puede configurar para que el acceso a la base de datos se realice a través de un proceso de autenticación, lo cual permite que solo los usuarios registrados en el sistema, puedan acceder a los datos correspondientes.

Servidor http

Un servidor HTTP, no es más que un software especializado que está constantemente atendiendo a un puerto de red determinado y en caso de haber alguna petición pendiente, asegurarse de manejarla de forma adecuada. Para hacerlo, existen muchísimos programas que se pueden utilizar, pero por las características mencionadas anteriormente, se seleccionó NodeJS como fuente para crear dicho servidor.

El servidor diseñado, implementa una API (Interfaz de Programación de Aplicaciones) [[20](#_ENREF_20)], que brinda varias rutas (endpoints) a través de los que se puede interactuar con el sistema, creando así una capa de abstracción sobre la que se pueden servir varios tipos de clientes, ya sean clientes web, como aplicaciones para PC o para dispositivos móviles. Cada una de estas rutas tiene una y solo una función específica, por ejemplo: Agregar un sensor al sistema, modificar un usuario, autenticarse en el sistema, registrar un nuevo usuario entre otras.

Para la conformación de las rutas de la API, se utilizó el módulo **express**, dado que no solo es el más popular, sino que es uno de los mejores de su tipo. Express posibilita el manejo de las rutas del servidor de forma sencilla, limpia y con una gran versatilidad a la hora de configurarlo, de hecho, uno de los más populares stacks entre los desarrolladores es el stack MEAN, para el conjunto MongoDB, Express, Angular y NodeJS. La gama de opciones que nos brinda es muy grande, facilitando la extensión de las funcionalidades a través de los llamados **middlewares** o intermediarios, que son funciones que ejecutan de forma secuencial, cada una transformando la petición que llega al servidor para un mejor manejo o interpretación, así como otras funcionalidades como la de subir archivos hacia el servidor, asegurar que un usuario está autenticado o que tiene el nivel de permisos suficientes para ejecutar una operación determinada.

Dentro de los aspectos a resaltar en cuanto a la seguridad, se utilizó el método conocido como **bcrypt** para crear un hash de las contraseñas una vez llegan al servidor, y guardar en la base de datos el hash generado para así proteger las contraseñas de los usuarios y lograr un nivel de seguridad adecuado, dado que **bcrypt** es uno de los métodos más seguros para generar hashes que son poco propensos a colisiones y son, hasta el momento, imposibles de realizar el proceso inverso. Para la autenticación en cuestión, se seleccionó el método de **autenticación mediante tokens**, o **JWT** (JSON Web Token) que no es más que una “llave de acceso” que el servidor le proporciona al usuario una vez se autentica en el servidor. Este método no es el más eficiente, existen otros mejores como el uso de las cookies, pero en el caso de una API, es más conveniente usar autenticación por tokens.

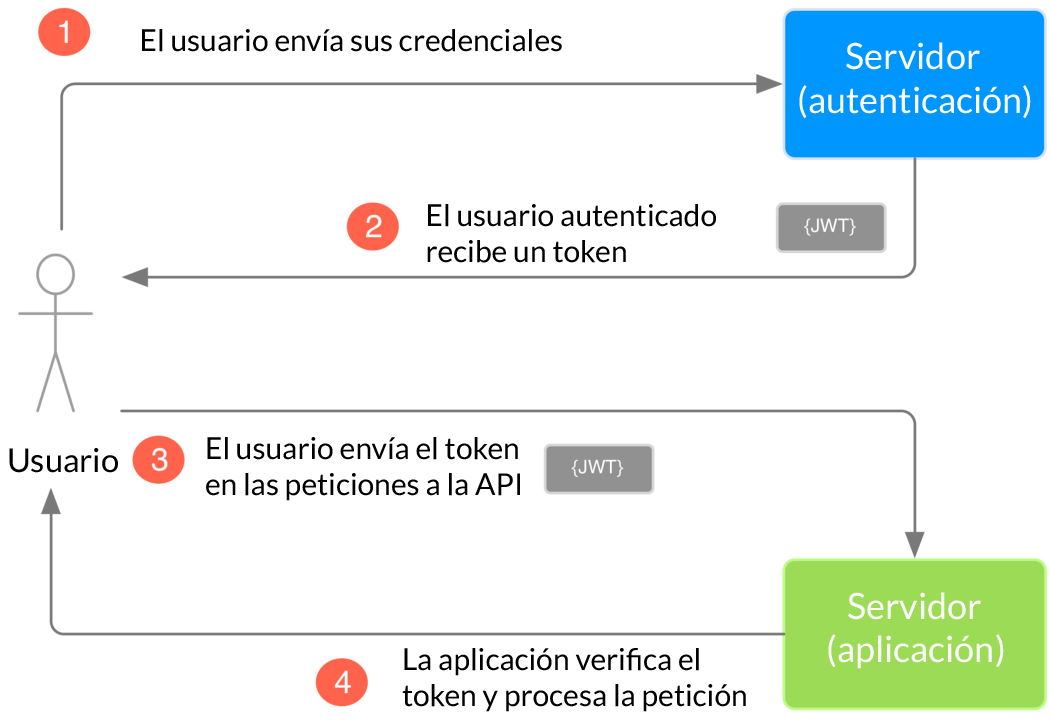


Figura 2. Proceso de autenticación por tokens

El uso de cookies es fácilmente implementado por los navegadores, dado que se envía como una cabecera HTTP dentro de la propia petición, sin embargo, no es implementado por todos los dispositivos, limitando el espectro de posibilidades a solo el cliente web. Además, requiere que se acceda a la base de datos o que se realice cierto procesamiento para validar su autenticidad, lo que puede llevar a que un sistema con una alta demanda, como es nuestro caso, pierda mucho tiempo validando los datos de la sesión enviados y que su desempeño no sea favorable. Por otra parte, los tokens tienen la forma **<metadatos>. <datos>. <firma>**, donde la última parte es el hash de la conjunción de los metadatos, los datos y una clave secreta que solo existe en el servidor, que es quien genera dichos tokens, de modo que, si alguien quiere realizar un ataque y cambia alguno de los datos, a la hora de comprobar su autenticidad, el hash generado va a ser diferente y por tanto se tomarán las medidas pertinentes. La Figura 2 muestra las etapas por las que transita el usuario, desde su autenticación hasta el consumo de los recursos de la API. Otra cuestión importante es que, si alguien logra robar el token de otra persona, solo lo podrá usar por un tiempo determinado, ya que dentro de los parámetros de dicho token, se incluye una fecha de expiración, que generalmente son 20 minutos luego de que el usuario se autenticó en la página. Pasado ese tiempo, tendrá que volver a autenticarse y un nuevo token será generado. Dadas estas características, se optó por el método de autenticación mediante tokens, debido a la seguridad que brindan y a la rapidez de comprobación sobre su validez, haciendo que la API sea a la vez rápida y segura.

Cliente web

Toda la estructura explicada anteriormente, no tiene mucho sentido si no existe una forma de interactuar con ella, y ahí es donde entra a jugar un papel de suma importancia el cliente web. Dicho cliente es una página web creada usando Angular, una plataforma muy popular y potente, dirigida al desarrollo de páginas conocidas como **SPA** (Single Page Application, o Aplicación de una sola página). La filosofía de Angular es proveer de las herramientas necesarias para desarrollar páginas web basadas en componentes, que tengan un alto rendimiento, preferentemente para páginas que tengan un nivel de complejidad medio o alto y con capacidad de funcionar a la perfección en navegadores antiguos, reimplementando las últimas características de JavaScript (Promesas, Programación reactiva, Funciones asíncronas etc....) para garantizar la compatibilidad con la mayoría de los terminales y navegadores posibles.

Aunque el código que se ejecuta en el navegador está escrito en JavaScript, los proyectos hechos en Angular son escritos en TypeScript [[21](#_ENREF_21)], que engloba las características de JavaScript, pero se pueden **anotar** las variables con un tipo de datos determinado, posee un manejo avanzado de la programación orientada a objetos, brindando la posibilidad de crear clases para modelar los componentes. Soporta características como la herencia, la sobrecarga de funciones, entre muchas otras bondades que no son fácilmente accesibles desde JavaScript. Para poder ejecutar todo ese código, primero se somete el mismo a un proceso conocido como **transpilación**, en el que se convierte de TypeScript a JavaScript, haciendo todo el trabajo pesado para usar todas las características nuevas presentes en TypeScript, en un entorno desde el cual no son accesibles por defecto.

Anteriormente se hizo referencia al uso de componentes. Un componente no es más que un conjunto de elementos, que permite modelar una parte específica de la página web de forma aislada del resto de los componentes que puedan existir. El componente tiene asociada una clase en TypeScript, una plantilla HTML y uno o más archivos de estilo, ya sea en CSS, SCSS u otros. Una de las ventajas fundamentales de usar este tipo de metodologías, es que se puede enriquecer la sintaxis rudimentaria de HTML, asociando las variables de la clase, con fragmentos del código, de forma tal que, si la variable de la clase se modifica, automáticamente se refleja el resultado en el código HTML correspondiente. Esto se conoce como **data binding** y se puede realizar en dirección de la clase al código HTML, viceversa, o en ambos sentidos. Otra diferencia que tiene gran relevancia respecto a la forma tradicional de operar en la web es que, en la programación tradicional, una vez que se incluyen varios archivos de estilo CSS, estos pueden aplicar diferentes estilos a un mismo elemento, de forma que el estilo que prevalece es el último que se aplica, lo que puede llevar a resultados no deseados, y la situación empeora cuando se tiene el proyecto estable y por alguna razón, se decide incluir otro nuevo archivo CSS al proyecto. Angular soluciona este problema con una característica llamada **view encapsulation** (encapsulación de las vistas), que automáticamente le agrega algunos “marcadores” al código CSS, de forma que dicho estilo solo afecte al componente para el que fue escrito y no para el resto, lo que conduce a la reutilización de los componentes en varios entornos, sin miedo a que su estilo se vea afectado por el código CSS de otro componente.

Una de las herramientas más relevantes que Angular provee, se llama **tslint**. Dicha herramienta tiene como propósito recorrer todo el código TypeScript y detectar tanto errores de sintaxis como errores semánticos, así como sugerir modificaciones para un mejor rendimiento o porque alguna función o característica, está depreciada y su uso no será soportado en futuras versiones. El uso de esta herramienta no solo tiene ese objetivo, sino que es muy importante para cuando se va a realizar el trabajo en equipo, dado que cada uno de los integrantes del equipo de desarrollo debe respetar el estilo impuesto por el tslint y así todo el proceso de desarrollo es bastante uniforme. También conlleva a que, si algún miembro del equipo necesita ayuda, no hallan pérdidas de tiempo para adaptarse a un nuevo estilo de programación, sino que todo el código, aunque sea escrito por personas diferentes, sea semejante al menos en estructura.

Existen otras alternativas para la confección del cliente web, como es el caso de React (creada y mantenida por Facebook), Vue, jQuery, etc.... pero al existir un previo conocimiento y experiencia sobre la base de Angular, se optó por su selección para crear la página.

La página web cuenta con varias partes que facilitan el manejo de la información.

**Pruebas unitarias**

Las pruebas unitarias constituyen un mecanismo para probar el correcto funcionamiento de una unidad de código. Esto sirve para asegurar que el fragmento del código analizado, se encuentre funcionando correctamente y eficientemente por separado. Para este proyecto, se escribieron varias pruebas para validar que la API estuviera funcionando como era esperado, realizando múltiples pruebas para cada una de las rutas, emulando en cada una, un escenario diferente, de modo que no hallan sorpresas una vez se eche a funcionar el sistema, y que, en caso de ocurrir errores, que sean lo menos inesperados posibles.



Figura 3. Mocha y Chai

Para la puesta en práctica de las pruebas, se utilizó una combinación de módulos de NodeJS bastante populares, los cuales se muestran en la Figura 3. El primer módulo se llama **Mocha**. Dicho módulo permite ejecutar las pruebas propiamente dichas, manejando de forma óptima todo el entorno de desarrollo necesario para llevar a cabo las mismas. El otro módulo es **Chai**, que no es más que un módulo especializado en aserciones, o sea, que permite comprobar si un valor cumple con los requisitos previstos, donde en caso contrario, lanza una excepción que hace que la prueba correspondiente, falle. Una de las ventajas que proporciona es una amplia variedad de funciones que permiten validar cualquier resultado que pueda venir como respuesta desde el servidor, así como su uso a través de una sintaxis con un alto nivel de verbosidad, lo que facilita la escritura y comprensión de dicha prueba.

**Root**

Dentro del sistema propuesto, se consideró la opción de crear un usuario especial llamado **root**. Este usuario tiene la característica de ser el único usuario que no posee restricción alguna a la hora de interactuar con el sistema, privilegio con que no cuentan el resto de los usuarios. La razón que está detrás de esta consideración, es solo con el propósito de poder cambiar ciertos comportamientos que puedan afectar el funcionamiento usual de la página, por ejemplo, si ocurrió algún error en el sistema y se eliminaron todos los administradores, o si por algún problema en específico, los administradores o moderadores no pueden acceder al sitio etc... Este usuario va a estar la mayor parte del tiempo inactivo, dado que este tipo de errores no suele ocurrir. En lo que respecta a su visibilidad, es virtualmente invisible para el resto de los usuarios, incluso para los administradores. Una de las primeras verificaciones que hace el sistema una vez comienza a funcionar es verificar que existe ese usuario y de no ser así, procede a su creación, de igual modo que si no hubiera ningún administrador. Dicho usuario no es imprescindible, pero puede ser útil en determinadas ocasiones debido a su nivel de acceso en el sistema.

**Perspectiva futura**

Como parte de la ampliación futura del sistema de monitoreo implementado, se planea añadir un método de registro de eventos donde se guarden las acciones que se han llevado a cabo durante el funcionamiento del servidor. Es de suma importancia implementar dicho método de registro dado que, si ocurre algún error, se puede proceder a analizar los eventos previos al error, teniendo una mejor panorámica del asunto. Además, pueden ocurrir otros eventos inesperados, como que algún usuario elimine o añada información incorrecta o irrelevante que pueda llevar a un mal funcionamiento, en cuyo caso se puede simplemente revisar el historial de eventos del sistema y tomar las acciones pertinentes para que todo continúe funcionando como se espera. Esto añade un nivel significativo de robustez al sistema, debido a la capacidad de manejo de fallos, ya sean esperados o inesperados.

Se desea incorporar un sistema de mapas para poder ver cómo se están comportando los diferentes parámetros de la señal, segmentados por área geográfica o por transmisor etc.... llevando a un análisis más fructífero de las condiciones reales a las que están sometidos los receptores.

**RESULTADOS**

Se implementó el servidor y el cliente web con el propósito de recibir, almacenar y analizar los parámetros resultantes del muestreo de la intensidad o potencia recibida en el extremo receptor, la relación señal al ruido y la razón de bit erróneo, cumpliendo así con el objetivo propuesto. El servidor fue sometido a múltiples pruebas unitarias y pasó exitosamente cada una de las 36 pruebas realizadas sobre el mismo, mostrando el comportamiento esperado ante múltiples escenarios.

**RECONOCIMIENTOS**

Los autores desean agradecer a los tutores del presente trabajo por su constante preocupación por el desempeño del proyecto, al colectivo de trabajadores de LACETEL los cuales siempre estuvieron en plena disposición de ayudarnos y formar parte de un equipo de desarrollo.

**Sobre los autores**

**Isaac Vega Rodríguez**, estudiante de la Universidad Tecnológica de La Habana, cursando el 4to año de la carrera de Ingeniería en Telecomunicaciones y Electrónica. Alumno ayudante de Programación I y II, así como de Cálculo III y IV. Forma parte del grupo de desarrolladores y entusiastas CubanTech, así como del grupo de artistas aficionados de la CUJAE.

**Centro de trabajo**

***LACETEL***, Instituto de Investigación y Desarrollo de Telecomunicaciones.

**Misión del centro de trabajo**

Desarrollar **soluciones** de alto valor de conocimiento agregado **a problemas** específicos **de interés gubernamental** y tributar a la **independencia tecnológica** en Telecomunicaciones.

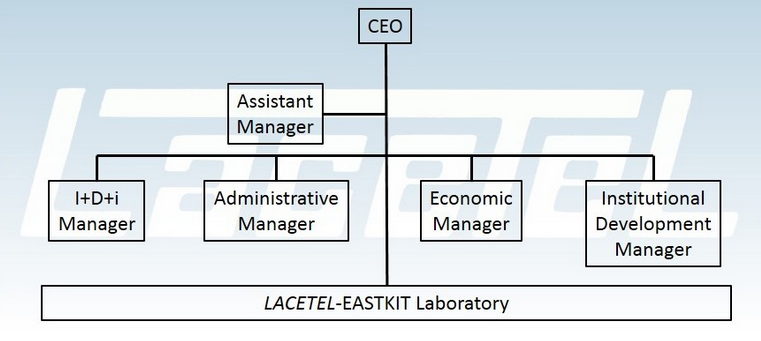
**Visión del centro de trabajo**

Ser una organización moderna y eficiente, cuyo **principal activo** es el **Talento, comprometido, motivado y con alto sentido de pertenencia**, líder en soluciones nacionales para telecomunicaciones y pilar del desarrollo cubano.

**Estructura organizativa**

Estructura plana, conformada por una dirección general y cuatro procesos (I+D+i, Logística, Economía y Desarrollo Institucional), así como un laboratorio en Shanghái que garantiza el soporte de materiales y contactos que garantizan mantenerse en el estado del arte, como se muestra en la Figura 4.

Figura 4. Estructura organizativa de LACETEL.



**Tutores**

**Rufino Cabrera Alvarez, Dreyelian Morejón Betancourt**

**REFERENCIAS**

1. Betancourt, D.M. and Y.H. Santana, *Calibración y ajuste de mediciones del nivel de potencia en herramientas de monitoren en DTMB basada en STB comercial.* 2018.

2. Santana, Y.H., et al. *DTMB monitoring tool based on a commercial set-top box*. in *2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. 2017. IEEE.

3. Valdes, A.I.M., et al., *Funcionalidades incorporadas al firmware del STB SDP160 para monitorizar la señal de TV digital.* 2018.

4. Valdes, A.I.M., et al., *Monitorización de la señal de televisión digital mediante el escaneo de canales utilizando la caja decodificadora SDP160.* 2018.

5. Calabrese, M., *Automated Frequency Coordination. An Established Tool for Modern Spectrum Management*, 2019.

6. Zhang, X., Y. Ma, and Y. Gao. *Blind Compressive Spectrum Sensing in Cognitive Internet of Things*. in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. 2017. IEEE.

7. Kulin, M., et al., *End-to-end learning from spectrum data: A deep learning approach for wireless signal identification in spectrum monitoring applications.* IEEE Access, 2018. **6**: p. 18484-18501.

8. Cohen, A.E., et al. *Radio Frequency IoT Sensors in Military Operations in a Smart City*. in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*. 2018. IEEE.

9. Lu, Q.N., et al. *State-of-the-art and challenges of Radio spectrum monitoring in borderlands of China*. in *2016 URSI Asia-Pacific Radio Science Conference (URSI AP-RASC)*. 2016. IEEE.

10. *SM200A — 20 GHz Real-time Spectrum Analyzer*. Available from: https://signalhound.com/products/sm200a-20-ghz-real-time-spectrum-analyzer/.

11. *Spetrum Monitoring and Interference Systems*. Available from: https://[www.keysight.com/en/pcx-x205197/spectrum-monitoring-and-interference-systems?nid=-32509.0&cc=CU&lc=eng](http://www.keysight.com/en/pcx-x205197/spectrum-monitoring-and-interference-systems?nid=-32509.0&cc=CU&lc=eng).

12. Allen, C. *What’s the Cost of Your Spectrum Monitoring Solution?* 2018; Available from: https://signalhound.com/news/whats-the-cost-of-your-spectrum-monitoring-solution/.

13. George, M., et al. *A compact multichannel data acquisition and processing system for IoT applications*. in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 2015. IEEE.

14. Dharur, S. and K. Swaminathan. *Efficient surveillance and monitoring using the ELK stack for IoT powered Smart Buildings*. in *2018 2nd International Conference on Inventive Systems and Control (ICISC)*. 2018. IEEE.

15. Pérez, D., et al., *Spectrum Forecasting Model For IoT Services.* 2018.

16. K.Moorthy, Y. and S. S. Pillai, *Wide Band Spectrum Sensing in Cognitive Radios using Compressed Sensing based on Improved Matching Pursuit Algorithms.* 2017.

17. Naqvi, S.N.Z., S. Yfantidou, and E. Zimányi, *Time series databases and influxdb.* Studienarbeit, Université Libre de Bruxelles, 2017.

18. Pasquali, S., *Mastering Node. js*. 2013: Packt Publishing Ltd.

19. Nayrolles, M., R. Gunasundaram, and S. Rao, *Expert Angular*. 2017: Packt Publishing Ltd.

20. Bojinov, V., *RESTful Web API Design with Node. js*. 2015: Packt Publishing Ltd.

21. Jansen, R.H., *Learning TypeScript*. 2015: Packt Publishing Ltd.