Revista digital de Semilleros de Investigación REDSI

SOFTWARE LIBRE APLICADO A LA TELEMEDICINA PARA ACTUALIZACIÓN DE INFORMACIÓN MÉDICA EN UNA PLATAFORMA DE ALMACENAMIENTO DE DATOS EN LÍNEA.

FREE SOFTWARE APPLIED TO TELEMEDICINE FOR UPDATING MEDICAL INFORMATION ON AN ONLINE DATA STORAGE PLATFORM.

[1]A. Cuevas
[1]Ingeniería Electrónica,
[1]Universidad de Pamplona

Resumen:

Los rápidos avances de la medicina y sus tecnologías generan gran cantidad de información necesaria que requiere ser almacenada y procesada, en conjunto con acceso interactivo de estos datos para facilitarle al personal médico poder hacer una valoración acertada a sus pacientes. Por consiguiente, se ha desarrollado una plataforma virtual en donde se lleve de forma interactiva el historial clínico de los pacientes de cada profesional de la salud en conjunto con una actualización y es que dicho sistema puede cargar señales unidimensionales e imágenes de mediana y baja resolución, almacenando esta información en una base de datos de tipo noSQL. En donde se establece una conexión bidireccional conocida como cliente-servidor en tiempo real mediante los *Frameworks* de NODE.js, en el cual se establece un servidor web que controla todas las peticiones que se realicen desde el *Frontend* al *Backend* y viceversa con un enlace a la base de datos. En vista del sistema presentado ejercerá una visualización interactiva de la información con el fin de facilitar y mejorar la atención sanitaria del personal médico con el paciente, obteniendo como resultado la implementación de un software que presente toda la información que se ha generado y que es necesaria a la hora de tratar un caso clínico.

Palabras clave: Telemedicina, Plataforma, Base de datos, diagnostico.

Summary:

The rapid advances of medicine in technology generate a great variety of important information that must be stored and processed, it is necessary to obtain an interactive access of such information so that the medical personnel can establish their diagnosis of

the patient's clinical case. Therefore, a virtual platform has been developed where the clinical history of the patients of each doctor is interactively taken together with an update. This system can load mono-dimensional signals and medium and low-resolution images, storing this information in a database of type noSQL. with a bi-directional connection between client and server is established in real time using the NODE.js Frameworks, in which a web server is established that controls all the requests made from the frontend to the backend and vice versa with a link to the database. In sight of the presented system it will exercise an interactive visualization of the information in order to facilitate and improve the health care of the medical staff with the patient, obtaining as a result the implementation of a software that presents all the information that has been generated and that is necessary at the time of treating a clinical case.

Key words: *Telemedicine, platform, database, diagnosis.*

I. Introducción

En el sector salud, trabajan en innovar su tecnología haciéndola un poco más eficiente a la hora de realizar servicios sanitarios, otorgando al personal médico aumentar su calidad, desempeño, eficiencia y efectividad de su trabajo disminuyendo las probabilidades de error a la hora de realizar el diagnóstico y el tratamiento de sus pacientes. Aunque la telemedicina avanza a grandes rasgos día a día se enfrenta con grandes riesgos, en especial con otorgar seguridad y confidencialidad a sus pacientes en cuanto a sus datos o historiales clínicos. Esta tecnología presenta grandes derivaciones, pero se hará énfasis en el almacenamiento de datos que son generados tanto por los médicos como por los dispositivos de biomedicina, en donde esta información va creciendo con el pasar del tiempo llevando a las entidades que implementan esta técnica optar por nuevas tecnologías conocidas como base de datos y ampliar la capacidad de los servidores, que son enlazados con aplicaciones como plataformas virtuales, aplicaciones móviles capaces de realizar un soporte para el personal de la salud e incluso realizar un autocuidado del paciente.

Se han desarrollado múltiples plataformas y aplicaciones móviles implementado técnicas para la atención sanitaria y brindando la seguridad de la información con es el caso de **saludtool** que es una plataforma colombiana muy utilizada por el personal médico para llevar sus historiales clínicos. Dichos softwares realizan un enlace bidireccional entre servidores [1] para poder transmitir imágenes, señales e información médica del paciente de forma virtual mediante plataformas de software. Con el fin de implementar la autenticación de usuario de forma segura otorgando obtener acceso a las bases de datos de forma segura, en el cual los autores [2] diseñan desde cero el protocolo TMIS (*telecare medical information system*), donde su funcionalidad es asistida por la nube para otorgar seguridad de teleasistencia, ya para mejorar la eficiencia del procesamiento de información se exponen la tecnología de cadena en bloques para poder generar mayor seguridad de información [9] en

cuanto a la contabilidad de las paginas, en caso de que el software no sea libre y tenga restricciones y se desee pagar por la función. En el ámbito de la telemedicina el Blockchain determina ciertos componentes como es el caso de árboles de Merkle para validad la información obtenida e incluso presenta la criptografía estableciendo hashes para identificar que las entradas del sistema sean las esperadas [7]. Además, se han definido nuevas técnicas para compartir imágenes en alta resolución estableciendo pizarras electrónicas y así definir el modelo pirámide de azulejos otorgando un procesamiento en tiempo real y reduciendo perdidas de resolución [4], proporcionando reducir un ancho de banda para lograr un mejor rendimiento computacional a la hora de realizar el registro personal de salud [5]. En otros casos se han desarrollados modelos que garantizan la disponibilidad y seguridad mediante un entorno móvil enlazados con el servidor, aplicando el conjunto de datos a algoritmos de IA para realizar un entorno de atención sanitaria efectivo [11], implementando clústeres de búsqueda distribuida capaz de generar diagnósticos por si mismos estableciendo la técnica de autocuidado mediante su dispositivo móvil para facilitar el diagnóstico médico y poder definir qué tipo de enfermedad puede estar padeciendo y/o establecer un mejor cuidado sanitario [12].

Teniendo en cuenta las estructuraciones nombradas anteriormente, se ha desarrollado un software libre basado en una plataforma asistida por una base de datos de tipo noSQL en donde procesa la información en formatos JSON (*JavaScript Object Notation*), la principal aplicación por la cual se generó este trabajo es basarse en llevar información del historial clínico que los pacientes tratan pero con mejoras, es decir, la ficha técnica podrá incluir tanto los datos médicos como imágenes y señales unidimensionales sintéticas seleccionadas de una base de datos virtual, con el fin de establecer que la plataforma está guardando y visualizando correctamente la información como sección de prueba ya que esta información es generada por los equipos biomédicos y los profesionales de la salud.

II. Arquitectura del sistema

Esta herramienta como su marco es basado en la web se divide en dos partes fundamentales para su desarrollo: la primera parte hace referencia al *Backend* el cual es donde se controla todas las peticiones que se le realicen a la plataforma para su navegación, en conjunto con la configuración de algunas bibliotecas que se presentan necesarias para el desarrollo del servidor web, con una técnica de comunicación con la base de datos. La segunda parte se trata del *Frontend* que es donde se diseña gráficamente la aplicación, es decir, el contacto directo que tendrá el usuario con el sistema para visualizar y cargar información e incluso iniciar sesión a su perfil personal para poder llevar el historial clínico de sus pacientes de forma privada. En la (fig. 1) se puede detallar gráficamente la estructura con la cual se está trabajando esta aplicación biomédica, en donde se establece la arquitectura de desarrollo para establecer todas herramientas que son requeridas a la hora la puesta en marcha o la producción del sistema. Se dice que la funcionalidad del backend se basa en controlar y establecer todas las funciones que entrega la plataforma al cliente, en conjunto

con el enrutamiento de las vistas o paginas que conforman al software para mostrar la información al cliente, es decir, que las vistas hacen parte del frontend donde se establece la interfaz grafica y es allí donde el cliente determina un contacto directo con la plataforma, realizando las peticiones que desee a la hora de navegar en la web. Dichas peticiones son establecidas por el controlador del software estableciendo una comunicación entre las dos partes mencionadas anteriormente, conocida como los métodos HTTP haciendo que el cliente le envié realizar una función determinada y el controlador le envié una respuesta que la puede visualizar el cliente, estas respuestas pueden ser mensajes de error en caso de que se ejecuto mal el evento, información que se encuentre dentro de la base de datos o incluso una afirmación de que el resultado fue satisfactorio. Estas peticiones son realizadas por el cliente mediante un evento "*onclik*" que puede contener información de un formulario y desee ser almacenada o validada por el sistema o simplemente un evento vacío como es el caso de cambiar de una vista a otra.

Backend

Esta sección se implementa mediante un entorno base de ejecución de JavaScript orientado a eventos asíncronos conocido como NODE.js, nos facilita un software que se debe instalar en el equipo que se está haciendo el desarrollo del sistema, con esta herramienta se instalará y procesará todas las librerías (*Express*, *Passport*, *Mongoose*, etc) que se necesitan tanto para la parte de producción como para la parte de desarrollo del software, ya que hay unas de ellas que nos facilitan realizar solamente el desarrollo del sistema, la instalación se realiza en una terminal integrada dentro de un folder a través del comando npm de formatos JSON. Primeramente, se empieza creando un servidor web local mediante Express.js es allí donde se configura el puerto de conexión a la web, la carpeta donde se encontrarán las vistas con un motor de plantilla y los documentos estáticos del sistema, los middlewares que configura y procesa el servidor y sus respectivas rutas de ejecución con la conexión a la base de datos. A continuación, se explicará detalladamente las bibliotecas más relevantes de su funcionamiento para producción dentro del servidor que se está configurando.

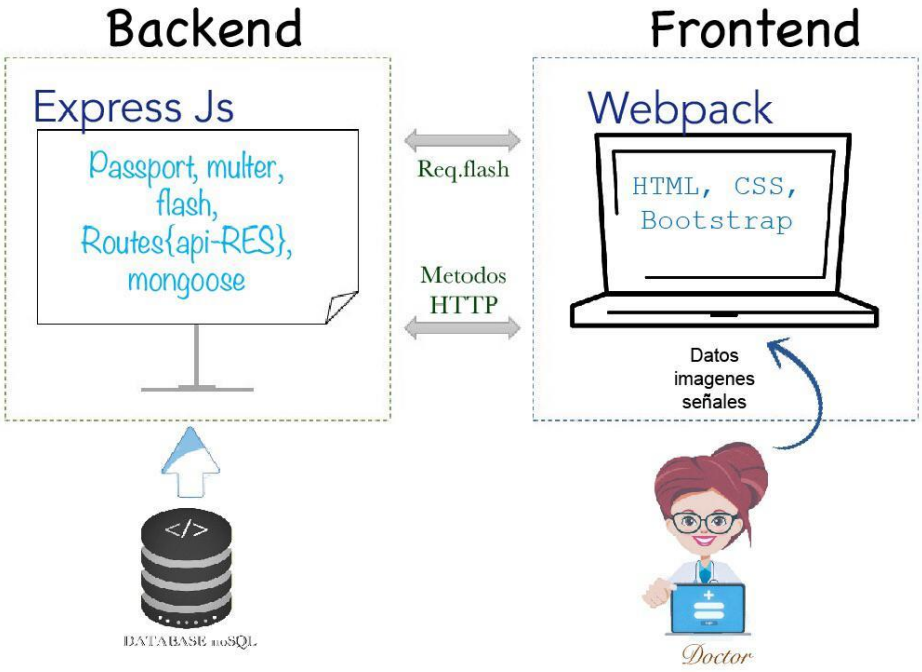

Fig. 1. Arquitectura del software gráficamente. (autor)

Mongoose.js

Se trata de un Mapeador de Documentos de Objetos (ODM) donde cumple una función específica y es establecer una conexión con nuestra base de datos mongoDB para poder almacenar y buscar información en ella, en este middleware es donde se puede definir objetos con esquemas tipiados, eso quiere decir, que podemos definir un modelo con los tipos de variables que se necesite para poder guardar la información, ya que su estructura está representada en formato JSON lo modelos creados con esta librería deben de contener todos los campos que se necesiten ya que no se pueden crear nuevas variables por parte del cliente y una vez definida le otorga funciones que nos permite guardar, eliminar, validar y consultar los datos que se encuentren guardados en mongoDB. Se crea un archivo .js que contiene la determinación de los modelos de los esquemas que se desarrolla y cuando se desee interactuar con la base de datos se le hace un requerimiento a este archivo para tener acceso a todas las opciones que fueron configuradas. Para realizar el enlace con la base de datos se le debe establecer una dirección a *mongoose* de donde se encuentran las credenciales, para la versión Atlas se establece una URL virtual y para la versión Server se determina la ruta donde se encuentre en el dispositivo de desarrollo.

Passport.js

Este módulo se trata de un middleware de autenticación de usuario ejecutado por NODEjs, el cual se encarga de administrar la autenticación mediante el método de sesiones controlando el inicio de sesión y la creación de una nueva cuenta, cabe aclarar de que se

desarrolla mediante la estrategia local por un correo electrónico y una contraseña. Esta función es requerida mediante los métodos HTTP definida como POST que se enlaza con esta librería haciendo el proceso de validación y creación de la información al cliente, si la respuesta a su validación es errónea envía un mensaje de alerta a través del módulo Flash a través de una solicitud a la ruta que se realizó la petición especificando cual ha sido su falla; en caso contrario si su validación es acertada retorna los datos del usuario y su estado autenticado mediante la función *Request* para que el código de estado de la petición lo tome como acertada.

También tiene un enlace con la base de datos para guardar y validad la información del usuario mediante un modelo de esquema denominado User. Pero existe una diferencia y es que la contraseña no es guardada en la base de datos originalmente, sino que se realiza una encriptación mediante el módulo *bcrypt-nodejs* en donde cifra y descifra la contraseña utilizando el método *hash* sincrónico donde incorpora un valor llamado *Salt* que permite establecer un fragmento aleatorio asociado con la contraseña y guardarlo en la base de datos. Evitando que dos contraseñas iguales generen el mismo hash en caso de que se presente un caso especial o incluso una extracción información por terceros al sistema

Multer.js

Esta función es definida para cargar imágenes al sistema en valor unitario, donde redirecciona las imágenes a una carpeta del servidor renombrando automáticamente el archivo dependiendo de la fecha y la hora a la cual ha sido procesada. Pero en este caso solamente se guarda en la base de datos la ruta en donde se encuentra esta imagen y su nuevo nombre con su extensión. Dentro del middleware se determina el nombre del archivo al cual va a recibir, haciendo que el formulario del *Frontend* sea de tipo *multipart/form-data* para que la ruta detecte que se está enviando una extensión File, para esta librería detecte que se está enviando un archivo de tipo imagen con el mismo nombre que se estableció y así poder activarse.

Rutas

Esta sección del servidor se encarga de controlar todas las peticiones que se realicen al sistema mediante los respectivos métodos HTTP para poder navegar por el sistema realizando renderizado de vistas, guardar, eliminar y editar información de la base de datos (CRUD) estas rutas son representadas mediante etiquetas virtuales conocidas como API-REST haciendo un enlace con el *Frontend* para cumplir una función en específico. Dentro de las funciones establecidas se trabaja con el método GET que se encarga de renderizar vistas del *Frontend* y realizar la petición de buscar información seleccionada en la base de datos para su visualización, el método POST se encarga de realizar peticiones de guardar lo que contenga un formulario en la base de datos, el método DELETE se encarga de realizar una petición de buscar un dato en específico mediante su ID y eliminarlo de la base de datos, por último, el método PUT que se encarga de editar un esquema en la base de datos

determinado dependiendo del formulario establecido en el *Frontend*. También se utiliza el método *Async await* con el propósito de que esa petición sea síncrona a la hora de interactuar con la base de datos, en otras palabras, lo que hace es realizar una pausa hasta que la base de datos nos devuelva la respuesta completa a la petición que hemos requerido.

Si estas rutas no son establecidas correctamente el servidor no va a funcionar correctamente y el software que se está diseñando no va a realizar lo que se desea que haga, es por esto que es necesario ser cuidadosos en esta parte haciendo que las etiquetas sean las mismas con las del *Frontend* y que se ejecute la petición que sea realizada. En fin, este apartado es definido como el motor fundamental para la funcionalidad de la plataforma a la hora de poner en producción o puesta en marcha el sistema.

Frontend

En este apartado es donde se estructura la comunicación directa entre cliente/servidor en un formato visual interactivo que pueda interpretar los datos fácilmente por el usuario. Se utiliza una librería de NODE.js que nos facilita el diseño la interfaz gráfica para relacionarla con la *Backend*, hacemos referencia a *Webpack* primeramente se utiliza porque nos brinda un servidor local independiente con el propósito de poder ir programando el archivo HTML e incluirle las animaciones o estilos que son proporcionados por el archivo CSS e ir de la mano con el *Framework* de diseño seleccionado que en este caso es *Bootstrap 4*, el servidor que genera dicha herramienta es muy útil ya que podemos separar las dos secciones principales que conforma la plataforma, pretendiendo de que los dos desarrollos no interfieran. A la hora del diseño de vistas se debe tener en cuenta de que las etiquetas que se han establecido en las rutas del *Backend* sean las mismas las que se establecen en los formularios, botones, opciones que se desarrollen en esta sección. A fin de que las peticiones que se realicen por el cliente presenten una comunicación fluida y sin errores con el servidor web, haciendo que la respuesta sea satisfactoria.

Fig. 2 desarrollo del frontend mediante la herramienta webpack (Autor)

Después haber diseñado todas las vistas necesarias, se configura ciertas opciones que necesita *Webpack* para redireccionar los archivos HTML, CSS Y JavaScript anteriormente diseñados a la *Backend*, dichas determinaciones hacen referencia a los *plugins* de la librería en donde se añade una herramienta adicional para enlistar los archivos html denominada *HtmlWebpackPlugin*, la cual nos ayuda a minimizar las líneas de código eliminando espacios innecesarios para concatenarlos en una sola línea, esto con el fin de no saturar el navegador donde se ejecute la aplicación. Por último, se debe enlistar los archivos CSS determinando el nombre de los archivos y su extensión que son configurados en las rutas API para que las vistas puedan ser leídas por el servidor web, y se ejecuta *webpack* en modo producción enviando los archivos a la *backend* donde se pueden ver reflejados en la carpeta pública del servidor haciendo automáticamente el llamado de los archivos de estilos dentro de las vistas.

También se tienen en cuenta la configuración de la herramienta *Chart.js* ya que su configuración se establece de parte del *frontend* y únicamente sobre la vista donde se desea realizar gráficas, dicha librería nos proporciona un archivo CDN virtual con el fin de tener acceso a todas las opciones que la componen. Como se desea graficar señales unidimensionales las gráficas serán de tipo lineal con escala automática en el eje Y, teniendo en cuenta de que las señales son sintéticas se debe establecer un límite de puntos en el eje X esto con el fin de que la gráfica no se deforme o pierda legibilidad de la información. Por defecto estas graficas se ven realizar dentro de un lienzo canvas para que los estilos que son configurados tengan un mejor contraste, haciendo que los colores y el área de la grafica tenga un renderizado suave para presentar mayor calidad visual.

III. Análisis de resultados

Los resultados se establecen en primera medida de la estructura visual o la interfaz gráfica que se ha diseñado para poder navegar en ella a través de la renderización de vistas y sus respectivas rutas controladas por el *Backend*, la segunda parte hace referencia a la forma como el usuario o el doctor puede ingresar los datos y/o señales al sistema y su forma de visualización e interpretación de los mismos para poder realizar el seguimiento y control del paciente al cual se le está llevando el historial clínico.

Para obtener un poco de conocimiento del funcionamiento de esta plataforma se hablará al respecto de todas las vistas que se muestran con sus respectivas secciones y sus funcionalidades, empezando por la vista principal que es como la portada de la plataforma, se visualiza su estructura en la figura 3 con cada una de los apartados que la componen.

Fig. 3 visualización de la vista principal (autor)

Teniendo en cuenta la semántica de HTML los primeros cuatro partes que visualizamos se encuentran dentro de un *navbar* o encabezado de página establecido por el *Framework de Bootstrap*, para darle forma y ubicación a los elementos que la componen: la primera hace referencia al logotipo de la página para darle un instintivo a la plataforma con su nombre y lema, el segundo instintivo se encuentra la barra de navegación la cual presenta una serie de botones enrutados para redireccionar al cliente a una sección en específico de la

misma, o ya sea para actualizar la vista y para dirigirse a una página virtual externa mediante su URL que se le ha establecido. El punto cinco se encuentra la información más relevante divida por secciones establecida o *article*, como decía anteriormente para conceptualizar el área a la cual se está ejecutando esta plataforma, exponiendo argumentos sobre la telemedicina y las bases de datos, una introducción a cerca de lo que se desea alcanzar si se utiliza este software y una pequeña conceptualización a cerca de la entidad que se encargó de su desarrollo, y la puntuación seis hace referencia al *footer* o pie de página en donde establece la privacidad de las cookies y derechos de autor con sus redes sociales.

Ya para entrar en materia de ejecución de la página, dentro de la vista que se muestra en la figura 3 aparecen dos botones; el primero lo que hace es redirigirlo a la vista de generar una nueva cuenta o registrarse a la plataforma (3) como se muestra en la figura 4, en donde se realiza una validación de que el correo electrónico no se encuentre registrado en la plataforma y la validación de confirmación de contraseñas sea iguales, de lo contrario el registro es aceptado. El segundo botón lo direcciona al inciso (4) de la imagen que es donde se inicia sesión, realiza la validación de que exista el usuario en la base de datos con el correo que está estipulando y que la contraseña sea la correcta. Si no presenta ningún error tanto para iniciar sesión como para registrase se redirecciona a la vista que aparece en la figura 4, haciendo la validación de que el usuario si se autentico correctamente y puede visualizar los datos y registros personales que ha generado y trabajar sobre el sistema.

Fig. 4 vistas de login y signup (Autor)

La figura 5 hace referencia al perfil del usuario en donde se muestra un encabezado (1) que visualiza el logo de la plataforma, la información que se ha generado cuando realiza el registro, como es el caso del nombre del usuario y el centro médico donde se está laborando

y un apartado final (1.1) que proporciona cerrar sesión. En esta vista se realiza una petición a la base de datos en tiempo real y es buscar todos historiales clínicos que ha generado este usuario y mostrar instintivos del paciente como es el nombre, el número de documento y lo visualiza en la sección (3) de esta vista mediante una tabla (3.1), esta búsqueda se realiza ya que mongoDB le genera un ID aleatorio cada vez que guarda información, en el caso de que guarde un historial clínico dentro del esquema se establece una forma de que almacene el ID del usuario que lo esté generando, esto se hace con el fin de que solamente visualice los datos personales del cliente y no poder ver la información de los demás usuarios. Dentro de la tabla hay una columna de acciones que se trata de establecer botones para eliminar el historial o editarlo. Cuando se edita el historial (3.2) lo que hace es mostrar una vista parecida a la que se renderiza cuando se desea crear una nueva ficha clínica, pero en este caso se visualizan todos los datos que contiene el historial ya generado donde permite editar y crear información del paciente. también se visualiza los anexos que son las imágenes que se han cargado y mostrar gráficamente las señales unidimensionales, en el caso de eliminar todo el historial de un paciente (3.3) lo que se hace es enviarle el ID del dato y pedirle al servidor que lo elimine ese esquema de la base de datos.

Fig. 5 vista principal del perfil del usuario para visualización de información que contiene. (Autor)

Siguiendo con el punto dos de la figura 5, lo que se crea es un menú con las opciones que otorga la plataforma como es el caso de la agenda médica (b), historial nuevo (c), soporte técnico (d). las vistas que se presentan dependiendo de la opción que se tome están

estipuladas en la figura 6, haciendo que las rutas rendericen y funcionen adecuadamente para la producción del sistema.

Cuando se selecciona la opción de realizar un historial nuevo, se renderizará la vista (1) que aparece en la figura 6, en el cual expone un formulario que contiene la plantilla de un historial clínico estándar que utilizan las plataformas de historiales clínicos, realizando la presentación de todos los casos médicos necesarios a la hora de llevar un control sanitario. dentro del apartado de revisión por sistema de la ficha técnica, se encuentran los anexos, es allí donde se puede cargar imágenes, nombre y datos de las señales biomédicas sintetizadas por los equipos médicos que se deseen almacenar; cabe aclarar de que los datos de estas señales deben de estar vectorizados para poderlos cargar, es decir, que la plataforma acepta solamente los datos numéricos separados por comas a la entrada del formulario, esta vectorización debe de contener un tamaño no mayor a diez mil datos y una amplitud con un rango que no supere las quinientas unidades entre ellos, en caso de no ser así no se podrán visualizar los datos de la señal y se deformaran o no serán legibles a la hora de ser graficadas.

Las otras opciones del menú son herramientas de valor agregado, con el fin que toda la plataforma sea un apoyo para el cliente que está trabajando con este aplicativo. Con es el caso de la agenda en donde se expone la ventana (2) de visualización de un calendario dinámico que le permitirá posicionarse en un día, mes o año que se requiera, después se expone un formulario que establece el título de la actividad que desea realizar, en conjunto con una fecha y hora, en la misma vista se visualiza las actividades que se han creado y poderlas eliminar cuando se hallan realizado o apetezca. Por último, se encuentra la opción de soporte técnico (3), le permite al cliente poder enviar un correo electrónico al proveedor de la plataforma realizando un mensaje con su ceja, petición o lo que necesite saber el cliente.

Fig. 6 vistas opcionales para el usuario (Autor)

En primera medida lo que el cliente debe conocer es la metodología de funcionamiento de la plataforma donde se debe empezar por crear una cuenta con sus datos personales (un usuario y una contraseña), insertar credenciales y entrar al sistema, una vez dentro cuando necesite crear un historial se dirige al menú de opciones nuevo historial, completa el formulario con la información del paciente generar sus anexos si es necesario de lo contrario se puede dejar los espacios vacío. Una vez generado el historial al paciente se podrá visualizar la información que se generó y se guardó en la base de datos en conjunto con los anexos establecidos, se deberá posicionar en la edición de los datos del paciente, es ahí donde se encontrara en formato visual y de forma ordenada dichos datos para poder presentarle al personal médico el objetivo fundamental del sistema (apoyo clínico), llevando el control del paciente con la información que se generó en la consulta. Si presenta problemas con el funcionamiento del software o el uso del mismo, se dirige a servicio técnico y con gusto se resuelve el inconveniente presentado, el propósito es brindar un apoyo confiable al cliente.

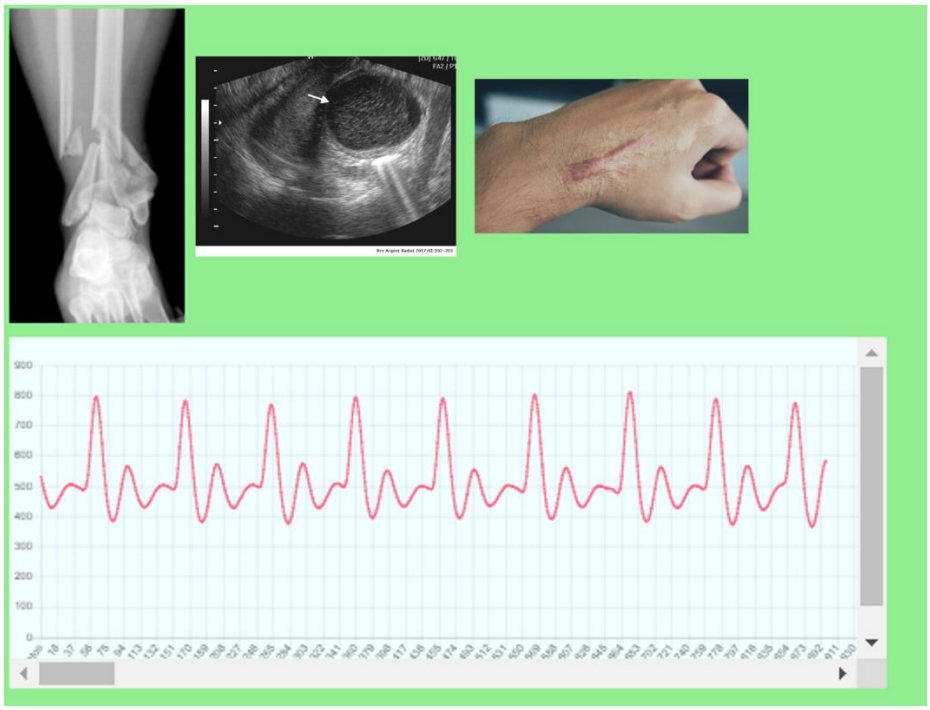

Fig. 7 visualización de anexos (Autor)

Presentando la actualización que se ha dispuesto a realizarle al historial clínico como actualización a los softwares convencionales, podemos visualizar en la fig. 7 se representan las señales médicas de forma visual, en donde se limita a cargar una sola imagen por paciente, como prueba se cargaron imágenes extraídas de la web con diferentes resoluciones para realizar pruebas de eficiencia al utilizar esta opción. para el caso de las señales unidimensionales se le da la posibilidad de cargar y visualizar con un tamaño menor o igual diez mil puntos, con una actualización automática en el eje Y, haciendo que el lienzo donde se visualiza la gráfica sea de forma dinámica haciendo que se actualice cada vez que se reciba un dato.

Como estrategia de mejorar el aplicativo, se busca implementar una comunicación en tiempo real entre el cliente y el servidor utilizando Framework de Node.js establecido como Serialport y Socket.io. La primera librería nos permite realizar una comunicación entre el dispositivo que está enviando la información serializada, permitiendo que el servidor pueda leerla y establecerla a la disposición del manejo de la plataforma, una vez obtenido el dato se desea enviarla a nuestros clientes para su visualización y cargarla a la base de datos, por lo tanto, se utiliza la segunda librería haciendo una conexión con el servidor para empezar a emitir los datos a los clientes. Para poder recibir los datos se debe establecer de que se esta enviando por una librería establecida mediante un encabezado y a la hora de renderizar la vista donde se realiza la configuración automáticamente se conecta y empieza a recibir los datos, donde se visualizan gráficamente utilizando la librería de Chart.js permitiendo recibir y visualizar en tiempo real la información que es enviada por el dispositivo electrónico.

IV. Conclusiones

En este documento, se estableció una metodología para el diseño de una plataforma de software sincronizada en la web para llevar a cabo el almacenamiento de datos médicos en una base de datos, esto con el fin de presentar una ficha del historial clínico al cliente para que pueda servirle de apoyo a la hora de llevar los casos de sus pacientes, ofreciendo poder cargar imágenes de mediana y alta resolución e incluso señales biomédicas sintetizadas en una dimensión, como por ejemplo una ECG, EMG entre otras. Mediante la herramienta de Node.js se desarrolló un algoritmo para poder crear un servidor web local, permitiendo que controle todas funciones que establece la plataforma, en conjunto con la implementación de la conexión de nuestro sistema con la base de datos mongoDB a través del ODM que nos facilita Node.JS. Esta base de datos como es de tipo noSQL y presenta una estructura de almacenamiento en formato JSON solamente se puede cargar información de tipo caracteres, pero como se están cargando archivos la plataforma solamente se guardara el nombre del archivo y la dirección donde se encuentre almacenado, ya que los redirecciona a una carpeta publica que contiene el servidor web donde almacena todas las imágenes que necesita la plataforma para sus vistas y además guarda las que el cliente ha subido como evidencia al historial clínico.

Además, se garantiza la funcionalidad del sistema y seguridad de información permitiendo que los datos de los pacientes del cliente sean resguardados para que no se vea afectada la confidencialidad de esta información o intimidad del paciente a fin de que no sean divulgados sin su autorización, además se le brinda un apoyo en cuanto al servicio técnico en caso de fallas se pueda resolver y seguir garantizando su funcionalidad. Con respecto a la base de datos, se es satisfactorio establecerla de tipo CRUD (*Create, Read, Update and Delete*) con el fin de que el profesional de la salud pueda actualizar los datos que contiene el paciente e incluso otorgarle al paciente el derecho de exigir editar los datos personales y/o adquirirlos en caso de que se realice la solicitud.

Por último, presentando los alcances que puede obtener la plataforma se determina que es posible aplicarse perfectamente obteniendo resultados satisfactorios para la realización de consultas y diagnóstico a los pacientes en general, pero hablando un poco de costos la mayoría de plataformas que implementan bases de datos como la que se ha realizado, con el pasar del tiempo se necesita mayor capacidad de almacenamiento y las *Backups* en las nubes virtuales son costosas e incluso la implementación de servidores hacen que la aplicación de estos software sean más costosos.

En conclusión, a la hora de cargar información en tiempo real a la plataforma se presentaron una serie de inconvenientes y es que al recibir los datos a frecuencias altas el navegador se satura o desborda haciendo que no reacciones correctamente y los datos no sean leídos correctamente, lo que conlleva a buscar otras técnicas de envió ya sea vectorizando la

señal cuando son emitidas por el dispositivo con el fin de realizar un encapsulamiento para que no se presenten pérdidas durante el proceso y el almacenamiento.

V. Trabajos futuros

De acuerdo con la arquitectura del sistema establecida en este documento, se puede implementar muchas más herramientas de la telemedicina para obtener una versión mejorada de esta plataforma, con es el caso de las teleconsultas realizando videoconferencias entre el médico y el paciente, también se podría establecer la conexión multiplataforma para realizar aplicaciones móviles garantizando una mejor seguridad de la información quizá utilizando las tecnologías del Blockchain, entre otro caso sería la conexiones remotas a dispositivos electrónicos mediante los protocolos de comunicación que garantiza Node.js con el propósito de cargar las lecturas en tiempo real a la plataforma y poder almacenarla. Sacando un poco más de provecho se da la oportunidad perfecta de implementar la inteligencia artificial ya que con el pasar del tiempo se va formando una Big data interesante para realizar entrenamientos, haciendo que se optimice los estudios realizados por el doctor, haciendo se realice de forma automática, quizá para predecir enfermedades, llevar un control autónomo del paciente, buscando una mejor eficiencia, seguridad y evolución de esta gran área biomédica como es la telemedicina.

Referencias

[1] Q. Zhang, "Medical data visual synchronization and information interaction using Internet-based graphics rendering and message-oriented streaming," *Informatics Med. Unlocked*, vol. 17, no. August, p. 100253, 2019.

[2] C. T. Li, D. H. Shih, and C. C. Wang, "Cloud-assisted mutual authentication and privacy preservation protocol for telecare medical information systems," *Comput. Methods Programs Biomed.*, vol. 157, pp. 191–203, 2018.

[3] C. Y. Wu, U. Iqbal, and Y. C. (Jack) Li, "Communication and diagnosis: Cornerstones for achieving precision medicine," *Comput. Methods Programs Biomed.*, vol. 157, p. A1, 2018.

[4] L. Qiao *et al.*, "Medical high-resolution image sharing and electronic whiteboard system: A pure-web-based system for accessing and discussing lossless original images in telemedicine," *Comput. Methods Programs Biomed.*, vol. 121, no. 2, pp. 77–91, 2015.

[5] J. Wang, M. Qiu, and B. Guo, "Enabling real-time information service on telehealth system over cloud-based big data platform," *J. Syst. Archit.*, vol. 72, pp. 69–79, 2017.

[6] M. M. Hassan, K. Lin, X. Yue, and J. Wan, "A multimedia healthcare data sharing approach through cloud-based body area network," *Futur. Gener. Comput. Syst.*, vol. 66, pp. 48–58, 2017.

[7] A. H. Mohsin *et al.*, "Blockchain authentication of network applications: Taxonomy, classification, capabilities, open challenges, motivations, recommendations and future

directions," *Comput. Stand. Interfaces*, vol. 64, no. October 2018, pp. 41–60, 2019.
[8] M. Chen and V. C. M. Leung, "Reprint of: From cloud-based communications to cognition-based communications: A computing perspective," *Comput. Commun.*, vol. 131, no. July, pp. 77–82, 2018.
[9] A. Farouk, A. Alahmadi, S. Ghose, and A. Mashatan, "Blockchain platform for industrial healthcare: Vision and future opportunities," *Comput. Commun.*, vol. 154, no. December 2019, pp. 223–235, 2020.
[10] M. Sharma and R. Sehrawat, "A hybrid multi-criteria decision-making method for cloud adoption: Evidence from the healthcare sector," *Technol. Soc.*, vol. 61, no. April, 2020.
[11] Y. Karaca, M. Moonis, Y. D. Zhang, and C. Gezgez, "Mobile cloud computing based stroke healthcare system," *Int. J. Inf. Manage.*, vol. 45, no. September 2018, pp. 250–261, 2019.
[12] W. Lin, W. Dou, Z. Zhou, and C. Liu, "A cloud-based framework for Home-diagnosis service over big medical data," *J. Syst. Softw.*, vol. 102, pp. 192–206, 2015.
[13] "expressjs · GitHub." [Online]. Available: https://github.com/expressjs. [Accessed: 15-May-2020].