UNIVERSIDAD DE ANTIOQUIA

DESARROLLO Y DESPLIEGUE DE UNA PLATAFORMA WEB PARA EL ALMACENAMIENTO Y ADMINISTRACIÓN DE BIOSEÑALES

Autor
Estiven Alejandro Rendón Escudero

Universidad de Antioquia

Ingeniería, Bioingeniería
Medellín, Colombia
2020

DESARROLLO Y DESPLIEGUE DE UNA PLATAFORMA WEB PARA EL ALMACENAMIENTO Y ADMINISTRACIÓN DE BIOSEÑALES

Estiven Alejandro Rendón Escudero

Trabajo final de semestre de industria presentado como requisito parcial para optar al título de:
Bioingeniero

Asesor:
John Fredy Ochoa, PhD.

Línea de Investigación:
Informática

Universidad de Antioquia
Ingeniería, Bioingeniería
Medellín, Colombia
2020.

Tabla de Contenido

DESARROLLO Y DESPLIEGUE DE UNA PLATAFORMA WEB PARA EL ALMACENAMIENTO Y ADMINISTRACIÓN DE BIOSEÑALES

1. RESUMEN

El propósito de este proyecto se centró en abordar el problema de la descentralización de los datos que se generan a través de la toma de muestra de bioseñales. En la actualidad es posible la adquisición de diferentes potenciales biológicos a través de diversos equipos electrónicos que cada vez se vuelven más precisos y más portables. Sin embargo, el almacenamiento y la disponibilidad de esta información sigue siendo un inconveniente ya que aún se presenta una descentralización de los datos con lo que no se puede mantener un control estricto sobre las señales adquiridas y sobre la información del paciente relacionado con dicha bioseñal.

Con el fin de solucionar este problema, se desarrolló una plataforma web para el almacenamiento y la gestión de bioseñales basada en tecnologías de JavaScript tales como Angular y NodeJS. Esta plataforma se enfocó en la centralización y la alta disponibilidad de los datos que son ingresados a ella. Para esto se utilizaron herramientas de almacenamiento en la nube tales como Mongo Atlas el cual es un cluster para bases de datos no relacionales y los servicios de almacenamiento Simple Storage Service (S3) de Amazon Web Services (AWS).

A través de la plataforma es posible registrarse como un nuevo usuario y después de obtener una nueva cuenta, se cuenta con la capacidad agregar un paciente nuevo a la base de datos con sus respectivas bioseñales, información que es almacenada en la nube. Es posible consultar la información de un paciente y visualizar los archivos con las señales asociadas a él, estas señales están disponibles para su descarga inmediata debido a que por cada paciente se hace una consulta en tiempo real a los servidores de AWS y solo mostrará en pantalla las bioseñales que estén estrictamente disponibles y relacionadas.

2. INTRODUCCIÓN

Gracias al avance continuo que ha tenido la tecnología en los últimos tiempos, ha sido posible llegar a un punto en el que procesar millones de datos es una tarea que toma un par de minutos cuando hace algunos años tomaría cientos de horas o inclusive sería imposible de realizar. Lo anterior, sumado a las diversas plataformas de almacenamiento en la nube y a los motores de búsqueda actuales, constituye una herramienta indispensable para la investigación en diversas ciencias. A pesar de esto, el desarrollo e investigación de muchas ciencias, incluidas la medicina y la ingeniería, han encontrado dificultades ya que el acceso a cierto tipo de información es limitado y en muchos casos la información no se encuentra debidamente documentada por lo que se hace imposible usarla para fines investigativos [1].

En la actualidad, es posible encontrar diversos repositorios de bioseñales en la web, para el caso de las señales EEG por ejemplo, se puede encontrar la base de datos del hospital universitario de Temple en donde se encuentran alrededor de 30.000 señales EEG tomadas a pacientes con diferentes patologías o el de la universidad de California en San Diego en donde también es posible encontrar cientos de muestras de EEG bajo diferentes condiciones [2][3]. Repositorios como estos han sido objeto de investigación en diversos estudios, sin embargo, la mayoría comparten algo en común, las bioseñales alojadas en estas páginas son de pacientes generalmente estadounidenses o europeos en algunos casos, lo que puede representar un gran sesgo en el resultado de algunos estudios ya que, en algunas investigaciones, la actividad cerebral, muscular o cardiaca de los individuos se podría ver influenciada por factores ambientales y genéticos. Teniendo en cuenta lo anterior, se ve la necesidad de poseer una base de datos de países en vías de desarrollo, como es el caso de Colombia, en donde es difícil encontrar información en la red sobre la actividad biológica de la población en sus diferentes regiones [4].

Debido a la situación planteada, se propone el desarrollo y despliegue de una plataforma web que emplee herramientas de almacenamiento en la nube con la cual se pueda crear un repositorio de señales biológicas que puedan llegar a ser usadas como base de datos para investigaciones relacionadas con la población colombiana. Inicialmente, la plataforma servirá para la administración y el almacenamiento de las bioseñales obtenidas por el grupo de investigación GRUNECO de la Universidad de Antioquia; después de realizar pruebas y poblar el repositorio, se tiene planeado liberar la base de datos al público usando un sistema de usuario y contraseña, para que dichos

datos puedan ser objeto de investigación. La aplicación web se realizará empleando HTML, CSS, Angular y Anguar Material UI para el desarrollo del frontend, Node.JS para el desarrollo del backend, una base de datos no relacional alojada en la nube a través de un cluster llamado MongoDB Atlas y los servicios de Simple Storage Service S3 de AWS para el almacenamiento de los archivos.

3. OBJETIVOS

Objetivo general

Desarrollar y desplegar una aplicación web la cual permita el correcto almacenamiento y administración de Bioseñales.

Objetivos específicos

- Diseñar una interfaz para la aplicación web y realizar la conexión con la base de datos y el sistema de almacenamiento
- Configurar y desplegar la aplicación web en un servidor y comprobar su disponibilidad a través de acceso remoto
- Validar el correcto funcionamiento de la aplicación web y poblarla con usuarios, bioseñales y su respectiva información

4. MARCO TEÓRICO

Las bioseñales son un medio de transmisión de información de un sistema fisiológico dentro del organismo humano, esta información puede dar indicios sobre el estado de dicho sistema a través del análisis de su frecuencia y amplitud. Las señales biológicas más estudiadas son las generadas por el cerebro, por el corazón y por los músculos. Gracias a la investigación de estas señales biológicas, ha sido posible conocer más sobre diversas patologías, sobre su temprano diagnóstico y sobre su tratamiento.

La electroencefalografía es una técnica que permite medir la actividad eléctrica cerebral mediante el uso de electrodos fijados sobre el cuero cabelludo. Esta técnica es usada en la medicina para diagnosticar trastornos cerebrales o tratar tumores cerebrales, daños cerebrales, encefalitis, epilepsia, entre otros [5].

La actividad cardiaca es medida a través de un electrocardiograma, el cual es una técnica empleada para amplificar y capturar la actividad eléctrica producida por el corazón, a través de ella es posible detectar actividad inusual del corazón tales como arritmias, isquemias y otros trastornos cardiovasculares [6].

La señal electromiográfica permite evaluar la salud de los músculos y de las neuronas motoras. Tras obtener esta señal es posible determinar si existen disfunciones nerviosas o musculares e inclusive determinar si existe algún problema con la comunicación entre los músculos y las neuronas motoras [7].

En los últimos años, gracias a la informática médica, el acceso a la información médica se ha facilitado lo que ha contribuido al desarrollo de nuevos proyectos investigativos y a diversos tratamientos. La informática médica se define como un conjunto de herramientas, técnicas y métodos que se emplean para manejar información médica. Este campo de la informática ha tomado un papel importante en la medicina ya que en la actualidad un gran porcentaje de información médica se gestiona y se procesa a través de aplicaciones de escritorio o aplicaciones web [7]. La adopción de esta tecnología en el campo médico se debe en gran parte a la facilidad que tiene el personal médico para agregar, consultar y modificar la información puesto que, gracias al correcto diseño de las interfaces, cualquier persona del personal médico con conocimientos básicos de ofimática, puede controlar el flujo de

información requerido en algún proceso médico, desde el manejo de historias clínicas, hasta el manejo de archivos como imágenes y/o señales médicas [8].

El desarrollo de estas herramientas ha sido posible gracias al desarrollo de diversos lenguajes de programación como Java, Javascript, Python, C++, PHP, entre otros y a diferentes frameworks de desarrollo como Angular, Springboot, Django, Laravel, entre otros. Estas tecnologías posibilitan el desarrollo de interfaces, la conexión con base de datos, la gestión de la misma y el despliegue de las aplicaciones en la web [9].

4.1. Frontend

El frontend es la parte de un programa o dispositivo a la que un usuario puede acceder directamente. Es un conjunto de herramientas que son implementadas para que el usuario tenga una buena experiencia dentro de la aplicación, una buena inmersión y usabilidad. La base para todo desarrollo frontend dentro de un navegador web, son los lenguajes de marcado HTML (Hyper Text Markup Language), CSS (Cascade StyleSheets) que permiten la construcción y estilizado de cualquier elemento visual, y también del lenguaje de programación JavaScript que brinda interactividad a la aplicación [10].

4.1.1. Javascript

JavaScript, más comúnmente conocido por sus siglas como JS, es un lenguaje de programación interpretado cuyo dialecto se basa en el estándar ECMAScript desarrollado por la compañía ECMA. Es un lenguaje orientado a objetos y basado en prototipos, es imperativo, débilmente tipado y adicionalmente es dinámico [11].

Fue desarrollado por Brendan Eich de Netscape y lanzado originalmente en el año 1995. En sus inicios se creó principalmente para ser usado del lado del cliente con el fin de darle fluidez y dinamismo a las páginas ya que hasta antes de su salida las interfaces web eran estáticas y únicamente estaban basadas en HTML y CSS. Este lenguaje fue diseñado con una sintaxis similar a C y adopta algunas características del lenguaje Java, sin embargo, se debe tener en cuenta que Java y JavaScript tienen sintaxis y propósitos diferentes [11].

Para interactuar con los elementos de una página web, JavaScript fue dotado con el Document Object Model o DOM el cual le brinda la capacidad de modificar, eliminar o agregar elementos según la interacción con el usuario.

Todos los navegadores modernos interpretan el código JavaScript integrado en el navegador y aunque en sus inicios se desarrolló pensado únicamente para realizar aplicaciones en el marco del cliente, la empresa Netscape Communications lanzó posteriormente una versión que se podía implementar del lado del servidor llamada Server-side JavaScript o SSJS con el fin de establecer una solución completa basada en el lenguaje JavaScript que pudiera realizar todas las interacciones tanto con el cliente como con el servidor [11].

En la actualidad JavaScript es uno de los lenguajes de programación más populares y demandados de la industria ya que ofrece una gran variedad de herramientas y permite hacer todo lo que se desee en la web. Debido a la popularidad de este lenguaje, se ha creado una comunidad a lo largo de todo el mundo que aportan cada vez más al crecimiento del lenguaje e inclusive empresas de alto calibre impulsan el desarrollo y el uso de estas herramientas. Gracias a este crecimiento, se han desarrollado diversas soluciones para el uso de este lenguaje de programación llamados frameworks, estos proveen un conjunto de características que facilitan el desarrollo de aplicaciones web ya que implementan funciones de seguridad, prácticas estandarizadas y librerías que dan un valor agregado muy fuerte [12].

Los frameworks del lado del cliente más populares de la actualidad son AngularJS desarrollado por la empresa Google, ReactJS desarrollado por Facebook y VueJS desarrollado por un ex empleado de Google. Para el lado del servidor, principalmente se usa el entorno Node.JS basado en JavaScript acompañado del framework ExpressJS [12].

4.1.2. Angular

Angular es un framework para aplicaciones web desarrollado en TypeScript, desarrollado y mantenido por Google y de código abierto, se utiliza principalmente para realizar aplicaciones web Single Page Aplication o SPA. Este tipo de aplicaciones consisten en la encapsulación de todas las interfaces en una sola página y mediante la interacción del usuario se van mostrando las interfaces necesarias sin necesidad de una recarga por completo del navegador [13].

Los desarrolladores de Angular tenían como objetivo el incremento de aplicaciones MVC ya que de esta forma se facilita el desarrollo de las aplicaciones y adicionalmente, se puede hacer un testing de la aplicación más completo.

El patrón MVC o Modelo Vista Controlador, consiste en la implementación e interacción de los tres componentes mencionados. La vista es el cuerpo del código que representa la interfaz de usuario o, dicho de otro modo, es todo aquello con lo que el usuario puede interactuar en la pantalla, generalmente las aplicaciones tienen varias pantallas y cada una de ellas hace parte del modelo el cual es la representación estructural de los datos de la aplicación. El controlador se puede considerar como un intermediario entre la vista y el modelo ya que es el encargado de captar la información que el cliente suministra tal como textos o clics en imágenes o botones, hace operaciones con dicha información y brinda una respuesta la cual es plasmada en el modelo. Este esquema se representa en figura 1 [13].

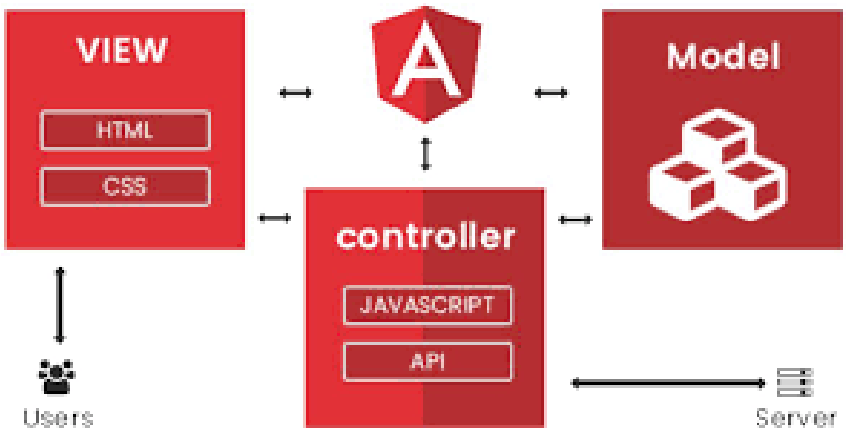

Figura 1. MVC en Angular.

Para crear aplicaciones con Angular, se generan templates con HTML los cuales son controlados a través de la lógica desarrollada en los componentes que deben ser exportados cómo clases. Del mismo modo se utiliza una arquitectura basada en servicios los cuales pueden ser inyectados en los componentes y así manejar la data y realizar todas las operaciones necesarias [14].

Para poder entender qué es Angular, es necesario comprender qué son los módulos. Un módulo es un contexto de compilación para un conjunto de componentes que son asociados con código y servicios con el fin de formar unidades funcionales. Ya que los desarrollos en Angular contienen varios módulos, existe un módulo root desde el cual se provee el mecanismo de arranque para la aplicación. Como buena práctica, se recomienda la creación

de varios módulos y organizar el código de manera funcional ya que esto permite el desarrollo de aplicaciones más robustas, más escalables y con un mejor rendimiento [14].

Además de los módulos, las aplicaciones de Angular tienen al menos un componente. Cada componente define una clase que tiene información y lógica y que está vinculada directamente a un template HTML y las directivas.

Un template es una mezcla entre lenguaje de marcado convencional y Angular markup cuyo fin es captar información y a través de la lógica de las directivas, definir lo que se va a desplegar o no en el DOM. Del mismo modo que se tiene un componente root en los módulos, también existe un componente root que permite crear una jerarquía con los componentes del DOM [13][14].

Finalmente, es necesario mencionar los servicios ya que constituyen toda la lógica que no está asociada directamente a una vista y que se quiere utilizar en diferentes partes de la aplicación. Estos servicios pueden ser agregados a un componente de la aplicación a través de la inyección de dependencias de Angular lo cual permite que solo entren a realizar las funciones para las que están desarrolladas, eso permite que las aplicaciones sean más ligeras y eficientes. En el esquema de la figura 2, se muestra el flujo dentro de una aplicación Angular.

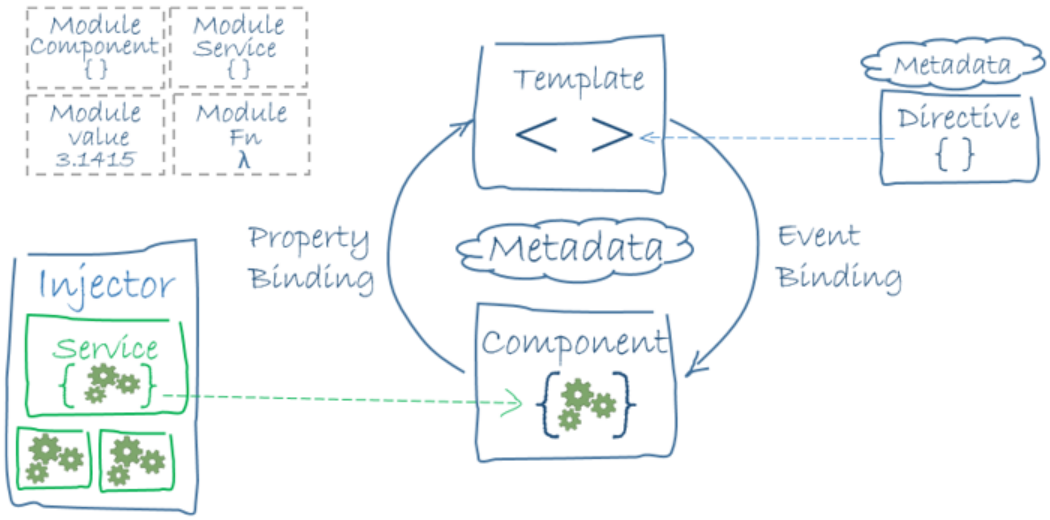

Figura 2. Flujo de una aplicación Angular

4.1.3. Angular Material UI

Angular material UI es una librería de componentes de interfaz con la cual es posible añadir ciertos componentes de una manera práctica, funcional y sencilla. Con el uso de esta herramienta, las aplicaciones web pueden verse más modernas ya que añade interactividad, estética y responsividad al producto final. Tiene múltiples componentes que pueden ser importados al componente principal de la aplicación y usados en cualquier módulo que pertenezca al proyecto. Cuenta con componentes tales cómo botones, tablas, menús desplegables, combo boxes, radio buttons, datepickers, toggles, steppers, toolbars, entre otros componentes que traen un estilizado por defecto sin embargo siempre es posible modificar estos estilos a través de CSS convencional o usarse en conjunto con componentes de otras librerías de componentes como Bootstrap [15].

4.2. Backend

El backend es la capa de acceso a datos de cualquier software o cualquier dispositivo. Esta capa no es accedida directamente por el usuario, sino que recibe información proporcionada por el usuario desde el frontend, hace un procesamiento o consulta de datos con dicha información y finalmente envía una respuesta con la información obtenida tras el procesamiento o consulta de datos. Este intercambio de datos entre el backend y el frontend es posible gracias al desarrollo de una API (acrónimo de Application Programming Interface). La API funciona del lado del backend y puede captar o enviar información a través de un formato XML o JSON [16].

4.2.1. Node.JS

Node es un entorno de ejecución multiplataforma, cuenta con código abierto, se usa principalmente para la capa del servidor, está basado en el lenguaje de programación JavaScript, es asíncrono, orientado a eventos y basado en el motor V8 desarrollado por Google. Fue creado por Ryan Dahl en el año 2009 con el propósito de crear programas de red altamente escalables y seguros [17].

Node funciona bajo un modelo de un único hilo de ejecución el cual tiene entradas y salidas asíncronas que pueden ser ejecutadas concurrentemente con un número de hasta cientos de miles sin incurrir en costos de rendimiento. Este enfoque de un solo hilo ejecución entre todas las solicitudes, atiende a necesidades de aplicaciones altamente concurrentes en donde cada operación sea de entrada o salida, requiere una respuesta o callback.

Esta herramienta cuenta con múltiples librerías para realizar operaciones con bases de datos relacionales y no relacionales de una forma sencilla y segura, adicionalmente provee lineamientos para blindar la aplicación a ataques maliciosos con los que se podría ver comprometida la integridad de los datos o de la aplicación misma, entre estos se encuentran:

- Librerías para encriptación de datos
- JWT para el manejo de sesión y autorizaciones
- Configuración de cabeceras HTTP
- Auditoría de módulos
- Análisis estático de base de datos

Además de lo anterior, Node cuenta con múltiples librerías para poder realizar conexiones a bases de datos en la nube como las proporcionadas por MongoDB Atlas el cual es un cluster en donde se pueden alojar bases de datos no relacionales de alta eficiencia o con servicios como los proporcionados por Amazon Web Services los cuales brindan un abanico casi ilimitado de posibilidades para el escalamiento y el crecimiento de cualquier aplicación [17].

4.2.2. Computación en la nube

Por otra parte, la computación en la nube se ha convertido en una herramienta importante dentro de la informática médica ya que ha facilitado el acceso desde cualquier lugar del mundo a información que antes solo era posible encontrar en alguna ubicación específica dentro de una bodega con cientos de cajas llenas de documentos impresos con los cuales se hacía tedioso realizar cualquier proceso investigativo. Las plataformas que han impulsado la computación en la nube han sido principalmente Amazon Web Services (AWS), Google cloud, Microsoft Azure e IBM cloud, cada una de estas plataformas ofrece un amplio abanico de servicios, entre los cuales se encuentran el almacenamiento de archivos, procesamiento de datos, herramientas analíticas y de inteligencia artificial, hosting, entre otras [18][19].

Para el caso específico de AWS, se cuenta con un sistema de almacenamiento llamado Simple Storage Service o S3, el cual permite el almacenamiento de prácticamente cualquier archivo que se desee, cuenta con una conexión sencilla a través de la librería SKD que es proporcionada por AWS y tiene

compatibilidad con múltiples entornos incluido NodeJS [19], a continuación, se muestra un esquema de la conexión entre S3 este entorno de desarrollo:

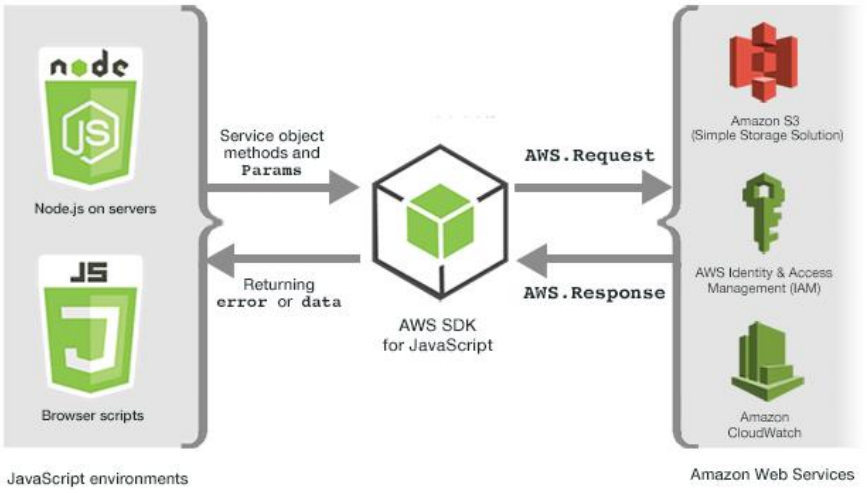

Figura 3. Intercambio de información a través de SKD.

Otra herramienta que se puede aprovechar es la plataforma MongoDB Atlas, el cual es un hosting de base de datos no relacionales alojado en la nube el cual ofrece múltiples opciones desde una capa gratuita hasta un servicio pago en el cual se puede tener almacenamiento sin límite y adicionalmente se puede acceder a un módulo con el cual hacer analítica sobre la actividad de los clusters alejados en ellas, en la figura 4 se muestra una ilustración que enseña todas las bondades del uso de esta plataforma [20].

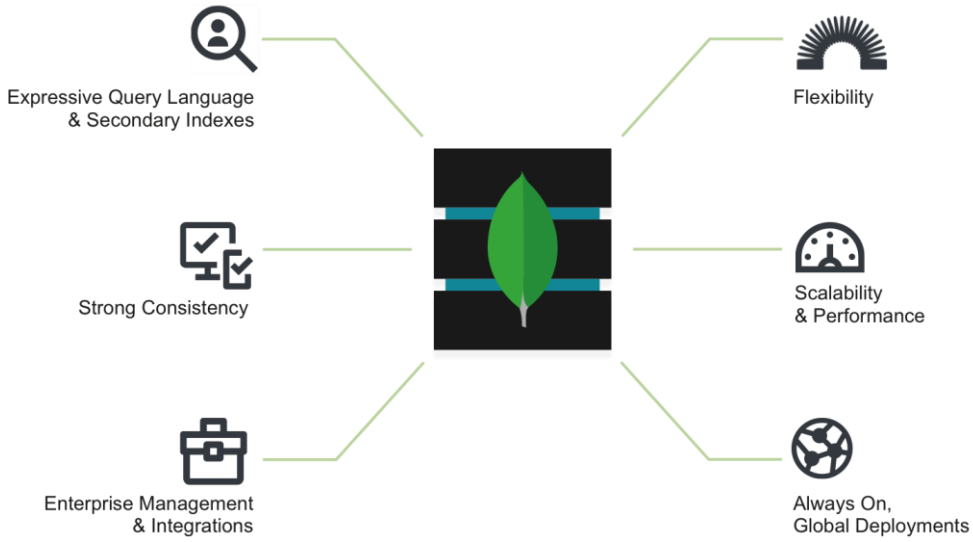

Figura 4. Características principales de MongoDB Atlas

La computación en la nube ha marcado un hito dentro de la investigación en los últimos tiempos ya que gracias al uso de alguna de las plataformas mencionadas anteriormente es posible almacenar grandes cantidades de información y además es posible procesar esta información a gran escala, con un alto rendimiento y a un bajo precio, todo esto ha facilitado el desarrollo y la planeación de nuevos proyectos de investigación ya que los costos y los tiempos de ejecución se pueden reducir significativamente ya que no se tendría que adquirir un computador de altas especificaciones o esperar a que alguna entidad propietaria de una de estas máquinas conceda el acceso por tiempo limitado [21].

5. METODOLOGÍA

Inicialmente se realizó una revisión bibliográfica con el fin de definir el conjunto de tecnologías que más convenian para la realización del proyecto con base en aspectos como la seguridad, la escalabilidad y sobre todo el soporte. Después de revisar exhaustivamente se eligieron tecnologías de JavaScript para el desarrollo de la aplicación, concretamente el framework Angular en su versión 9 para el desarrollo del frontend de la aplicación y NodeJS en su versión 12.18.2 como entorno para el desarrollo del backend. Con respecto a la base de datos, se empleó una base de datos no relacional MongoDB alojada en un cluster en la nube del proveedor MongoDB Atlas y finalmente, se usó el servicio de almacenamiento en la nube S3 de Amazon Web Services para el almacenamiento de las bioseñales.

Después de que se eligieron las tecnologías con las cuales se realizó el proyecto, se procedió con la definición del modelo de la base de datos no relacional, esto se realizó en conjunto con integrantes del grupo de investigación GRUNECO con el fin de que el proyecto fuera útil para ellos una vez terminado. Luego de obtener este modelo se procedió a configurar un cluster en la plataforma MongoDB Atlas, esta configuración consistió en la implementación de ciertos sistemas de seguridad para blindar el acceso a la base de datos por terceros y adicionalmente para obtener la cadena de conexión la cual permite el acceso a esta base de datos alojada en la nube siempre y cuando se cumplan todas las medidas de seguridad preconfiguradas.

Siguiendo con las configuraciones iniciales, se creó una cuenta en AWS y se realizó la configuración de la herramienta Simple Storage Service (S3) y se creó una instancia que fue dedicada exclusivamente al almacenamiento de las señales biológicas. Para el correcto uso de esta herramienta, fue necesario generar credenciales y claves de acceso desde la consola de AWS con el fin de otorgar acceso remoto a la aplicación.

El siguiente paso fue el desarrollo del frontend, para esto, se usó el framework Angular el cual brinda la posibilidad de realizar aplicaciones SPA, esto significa que es posible realizar navegaciones dentro de la aplicación o consumos de API’s sin necesidad de recargar la página lo cual brinda una mejor experiencia y una mayor velocidad a la hora de navegar a través de los diferentes componentes de la aplicación. En este paso se desarrollaron las diversas interfaces por las que un usuario de la plataforma podría navegar:

- Página de registro
- Página de ingreso
- Página Home
- Página de ingreso de nuevo paciente
- Página de ingreso de nueva bioseñal
- Página de búsqueda de pacientes
- Página de búsqueda de bioseñales
- Página de administrador

Estas interfaces fueron desarrolladas partiendo de herramientas simples tales como HTML y CSS así también como herramientas y librerías propias del framework Angular tales como RxJs que ayuda al comportamiento asíncrono de la página o de Angular Material UI la cual es una librería que ayuda con la generación y estilizado de componentes cómo botones o tablas.

Con las interfaces desarrolladas y establecido un modelo de base de datos, se procedió con el desarrollo del backend, este fue desarrollado utilizando en el entorno de desarrollo NodeJS basado en JavaScript y se utilizó el framework Express que proporciona métodos y herramientas para un mejor manejo de este entorno. Esta etapa del proceso consistió en la creación de diferentes APIs cuyo propósito es captar la información que llega desde el frontend, realizar operaciones y consultas a la base de datos y posteriormente dar una respuesta hacia el frontend con base en la información de entrada.

Otras funciones desarrolladas en el backend, consistieron en la implementación de medidas de seguridad para proteger la integridad de la información almacenada en la plataforma. Una de las principales medidas fue la encriptación de contraseñas a través de la librería Bcrypt, la cual ofrece un algoritmo de encriptación basada en iteraciones y en una clave secreta definida por cada usuario lo cual permite que sea casi imposible descifrar una contraseña a través de ataques de fuerza bruta o métodos similares.

Otra medida de seguridad implementada fue el uso de Jason Web Tokens o JWT, se usa principalmente para manejar las sesiones de los usuarios una vez son autenticados en la plataforma y consiste en la generación de una firma digital encriptada que contiene datos relevantes sobre el usuario y adicionalmente, cuenta con un tiempo de expiración, de esta forma se verifica que el usuario tenga una sesión válida cada vez que quiera realizar una operación después de ingresar a la plataforma. Finalmente, también se

desarrolló la integración de la librería SDK la cual permite conectar e interactuar con los servicios de AWS, para este caso, la implementación fue la carga, descarga y búsqueda de archivos en el bucket de S3.

Por último, se realizó el despliegue de la aplicación en la plataforma AWS, esta plataforma ofrecía una capa gratuita en la cual se pudo realizar la carga de la aplicación y posteriormente su despliegue; al ser una capa gratuita, la aplicación proporcionó un enlace genérico con el cual se pudo acceder a la aplicación y comprobar su correcto funcionamiento. En la figura 5 se muestra un diagrama de la metodología seguida durante el proyecto.

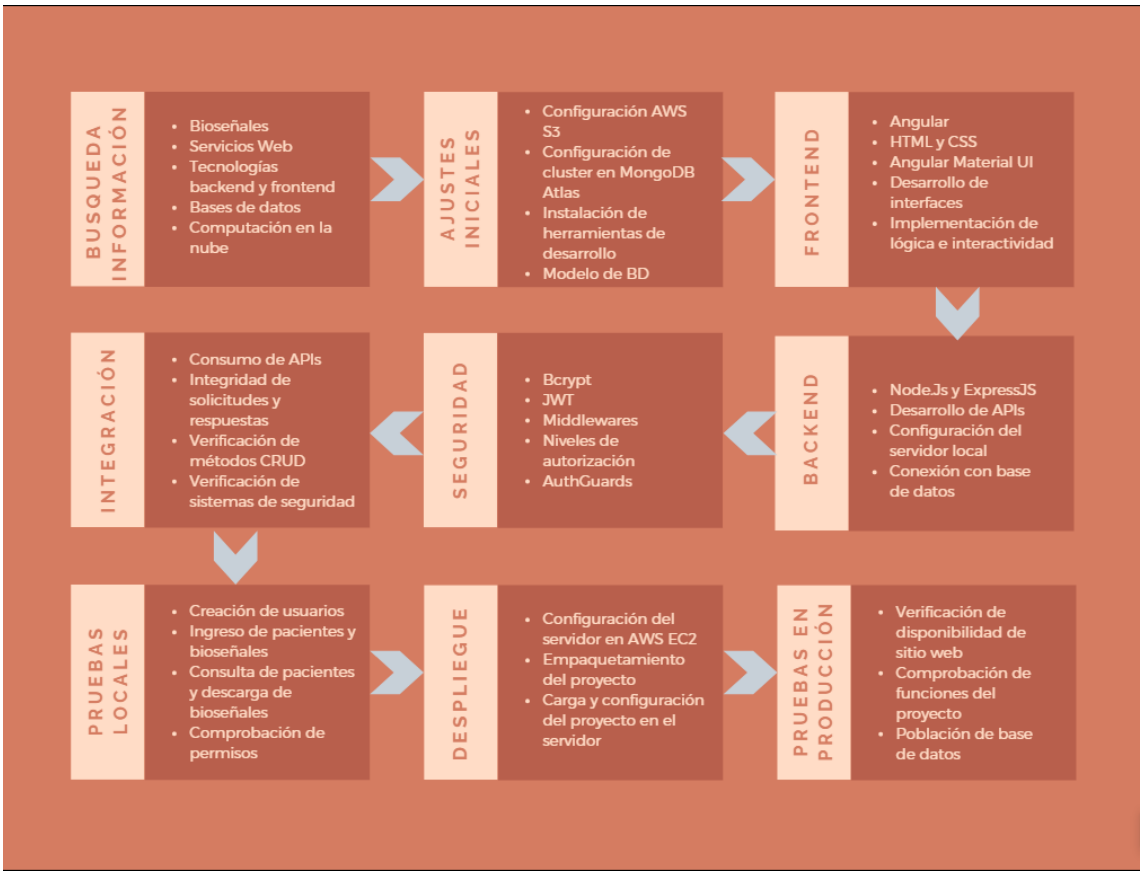

Figura 5. Metodología del proyecto

6. RESULTADOS Y ANÁLISIS

6.1. Esquema de base de datos

Después de realizar un acercamiento con los integrantes del grupo GRUNECO, se definió un modelo de base de datos no relacional que contiene la información más importante del paciente, dicha información es relacionada con su condición y con los intereses del estudio. Esta información se detalla a continuación:

- **Id**: Número único de paciente que es generado automáticamente por la base de datos cuando se ingresa el paciente
- **Label**: Este identificador es único para cada paciente y es definido por el técnico que tomó la muestra previamente
- **Age**: Se refiere a la edad actual del paciente
- **Gender:** Género con el que se identifica el paciente
- **Scholarship:** Nivel de formación académica del paciente
- **Ocupation:** Se refiere al oficio o profesión del paciente
- **Age_dcl:** En este campo se debe diligenciar la edad en la cual el paciente empezó a presentar síntomas de desorden neurológico
- **Age_dementia:** Se refiere la edad en la que el paciente fue diagnosticado con demencia, en caso de no tener dicha condición, se pone NA
- **Records:** Es la ruta dentro del servidor en el cual se alojan momentáneamente los archivos mientras son cargados en la nube, después de esto, el archivo se elimina por lo que posteriormente este campo puede ignorarse y no es mostrado en la interfaz
- **Content:** Campo de observaciones, se puede ingresar cualquier texto que el usuario de la plataforma considere necesario y que no esté contenido en los campos anteriores
- **Creator:** En este campo se inserta automáticamente el identificador único del usuario que está realizando el ingreso del nuevo paciente.

Esta definición del modelo de paciente no es definitiva y se puede redefinir dependiendo las necesidades de los usuarios de la plataforma. Esta es una de las ventajas con las que cuentan los aplicativos web, siempre están abiertos a cambios y actualizaciones a medida que las necesidades vayan evolucionando. A continuación, en la figura 6 se muestra la definición del modelo en Angular el cual contiene cada uno de los datos y su respectivo tipo de variable y en la

figura 7 se muestra un ejemplo de un paciente que ya fue ingresado a la plataforma.

```typescript
export interface Paciente {
  id: string;
  label: string;
  age: number;
  gender: string;
  scholarship: string;
  ocupation: string;
  age_dcl: number;
  age_dementia: number;
  records: string;
  content: string;
  creator: string;
}
```

Figura 6. Definición del modelo de paciente


```json
{
    "_id":{"$oid":"5ee00f6407f4a30b1461d298"},
    "label":"SUB002EMG",
    "age":"25",
    "gender":"F",
    "scholarship":"INGENIERA",
    "ocupation":"ASPIRANTE A MAESTRÍA",
    "age_dcl":"22","age_dementia":"25",
    "content":"NA",
    "records":"http://localhost:3000/images/SUB002EMG.rar",
    "creator":{"$oid":"5eb07123e4083c25944208a3"}
}
```


Figura 7. Ejemplo de un paciente ingresado en la base de datos

6.2. Características y navegación en la plataforma

Una de las características principales de cualquier plataforma web, es el sistema de registro y autenticación, estos sistemas deben estar dotados diversas verificaciones con las cuales se garantice la seguridad de la aplicación y se restrinja el ingreso de usuarios no deseados. Para el registro de un nuevo usuario en la aplicación, se desarrolló la siguiente interfaz mostrada en la figura 8.

Figura 8. Interfaz de registro de nuevo usuario

Durante el registro de un usuario, se realizan las siguientes validaciones y medidas de seguridad sin las cuales no se puede completar la creación de un nuevo usuario.

- **Validación de usuario existente**: En el momento en el que un usuario se va a registrar, debe ingresar un correo el cual debe ser único en la base de datos, en caso de que este correo ya exista, saldrá un mensaje de error, se rechazará la solicitud de creacion de nuevo usuario y se regresará a la página de registro para que la persona ingrese un correo que no exista dentro de la plataforma

Figura 9. Error de usuario ya registrado

- **Validación de formato de correo y de contraseña:** Ya que esta aplicación se desarrolló principalmente para el uso del grupo de investigación GRUNECO de la Universidad de Antioquia, se implementó una validación de correo electrónico con la cual se garantiza que el usuario pertenezca a la universidad. Por el lado de la contraseña, se estableció la regla de que debe contener al menos 8 caracteres, una letra mayúscula, una minúscula, un número y un carácter especial, todo esto con el fin de incrementar la seguridad de la contraseña y evitar que ataques cibernéticos de fuerza bruta, tengan éxito al intentar acceder a la plataforma. Finalmente se agrega el campo de verificación de contraseña para evitar que el usuario ingrese una contraseña equivocada y pueda seguir teniendo acceso a su cuenta. En caso de que la información ingresada no cumpla con los parámetros mencionados anteriormente, la aplicación no permitirá que se continúe con la creación de la cuenta.

Figura 10. Validaciones de ingreso de nuevo usuario

- **Encriptación de contraseña en base de datos:** A pesar de que ya se definió una directriz con respecto a la seguridad de la contraseña, en ocasiones alguna persona indeseada podría tener acceso a la base de datos y desde allí sustraer la información de cuentas o de pacientes. Para prevenir esto, se implementó la encriptación de la contraseña a través de la herramienta Bcrypt, esta herramienta realiza una encriptación usando un algoritmo que utiliza un número de iteraciones y una clave secreta que son definidas por el desarrollador de la plataforma. El algoritmo que usa esta herramienta, tiene la ventaja de ser adaptativo, es decir, con el tiempo las iteraciones pueden ir cambiando para volver más lento un ataque de fuerza bruta, inclusive este proviene de una herramienta con alto poder computacional. Por ejemplo, si un usuario ingresa la contraseña "**Medellin2020****", la contraseña que se guarda en la base de datos queda de la siguiente forma después de ser encriptada:

  "**$2b$10$kJKDfK1u03afGRNus7RsxOcZ6kNbqbXOSf0Vl5FH7xFuheTBVwlgS**"

  Cómo se puede observar, no existe una relación directa entre la contraseña original y la que se guarda en la base de datos, para poder decifrar esta contraseña sería necesario obtener el algoritmo que una la herramienta Bcrypt, conocer el número de iteraciones que maneja la plataforma y adicionalmente conocer la clave secreta que estableció el desarrollador al crear la plataforma. Todo esto brinda una mayor confiabilidad de los datos y los usuarios que serán ingresados en la base de datos.

Después de lo anterior, se creará un usuario nuevo, se guardará en la base de datos y saldrá un mensaje de que el usuario fue creado exitosamente. Ahora, el usuario puede autenticarse e ingresar a la plataforma, para esto se desarrolló la siguiente interfaz:

Figura 11. Interfaz de autenticación

Para esta sección también se implementaron ciertas validaciones y manejo de errores:

- **Validación de usuario existente:** Esta es la más obvia de las validaciones, el correo del usuario debe existir y además la contraseña debe pasar por un proceso de validación en el que es encriptada para poder ser comparada con la existente en la base de datos, en caso de que no se encuentre el usuario o la contraseña no coincida, se genera un mensaje de error.

Figura 12. Validación de usuario existente

- **Manejo de sesión a través de JWT:** Una vez se verifican las credenciales del usuario, se accede a la plataforma y se muestra la interfaz “home, en donde se tiene información básica de la aplicación e información de contacto, la forma con la que se logra mantener una sesión válida al interior de la aplicación, es a través de un token generado por la librería JWT de Node.JS, esta librería genera un string encriptado el cual es alojado directamente en el almacenamiento del navegador, este string contiene la siguiente información:

  - Un token único por cada sesión
  - ID del usuario
  - Rol del usuario
  - Fecha de caducidad del token

  Con esta información el usuario puede navegar a través de la plataforma según se lo permitan sus permisos y una vez se cumpla el tiempo de caducidad del token, el usuario es automaticamente deslogueado de la plataforma y devuelto a la página de autenticación.

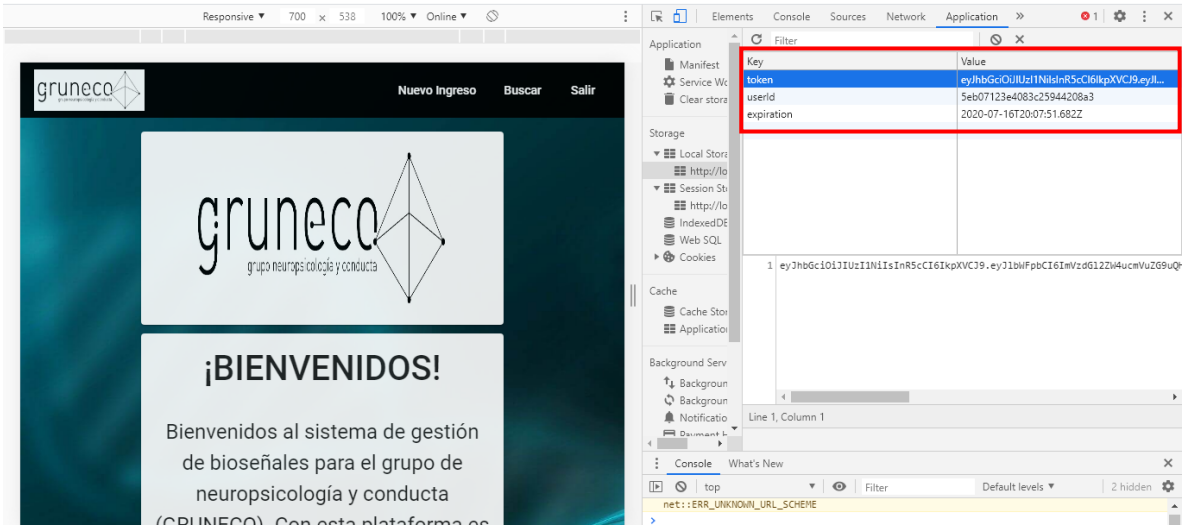

Figura 13. Token almacenado después de realizar autenticación

Una vez se haya realizado la autenticación, el usuario tendrá la posibilidad de realizar un nuevo ingreso, en esta sección se cuentan con dos opciones, una de ellas es ingresar un nuevo usuario con todos los datos establecidos por el

modelo o puede subir un paquete de bioseñales que puede estar o no asociada a un paciente. Para la sección de nuevo ingreso se desarrollaron las siguientes interfaces:

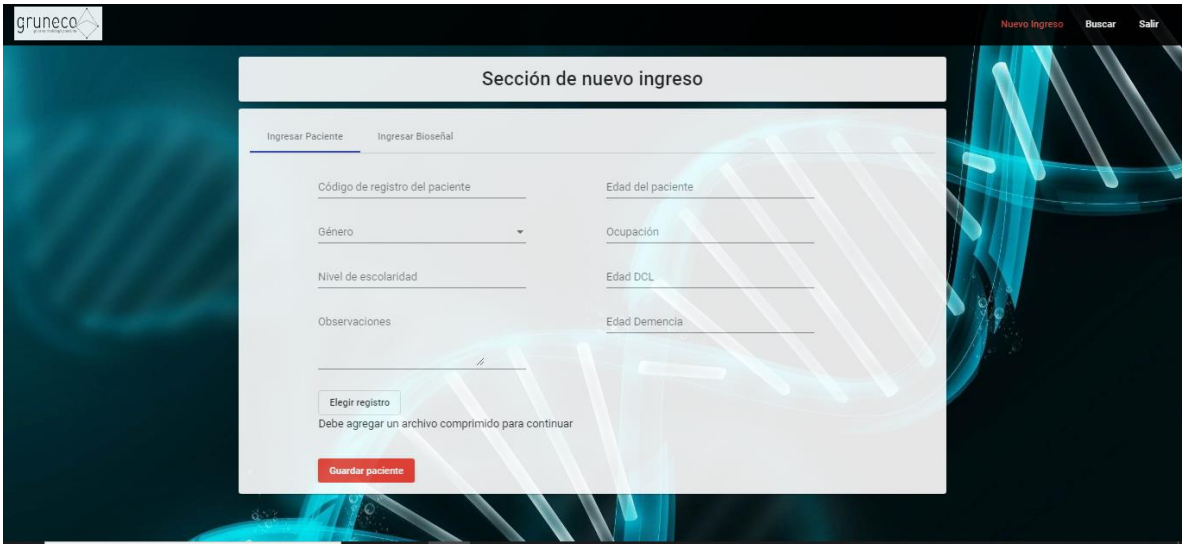

Figura 14. Interfaz de ingreso de nuevo usuario

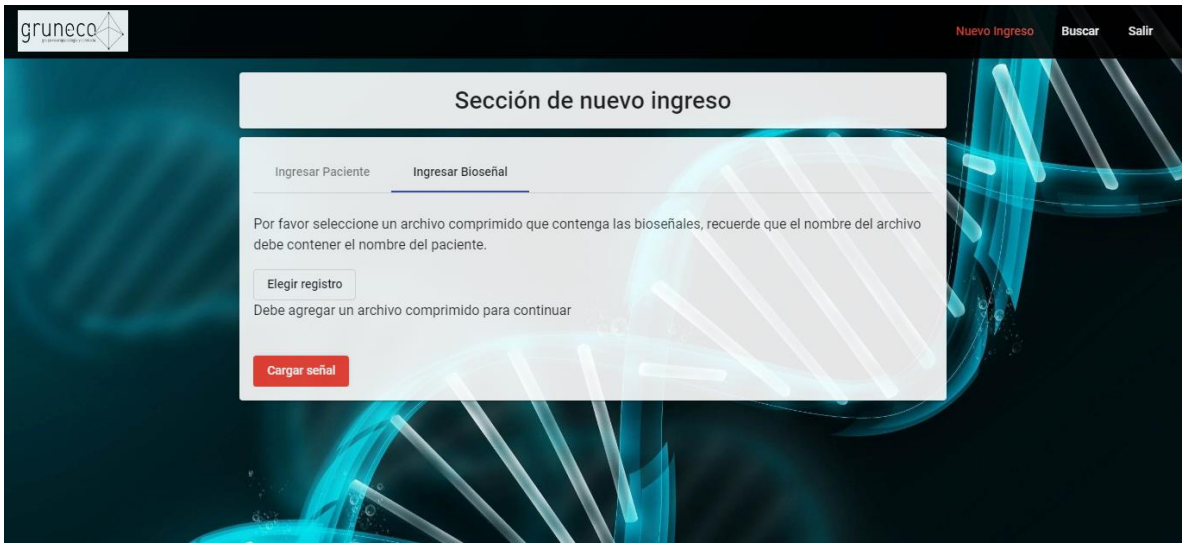

Figura 15. Interfaz de ingreso de nueva señal

En el momento de ingresar un nuevo paciente o una bioseñal, se tienen las siguientes consideraciones:

- El número de registro de usuario debe ser único, en caso de que el usuario ya exista dentro de la base de datos, se generará un error y no se podrá guardar en la base de datos.

- El usuario debe tener los permisos necesarios para ingresar a un nuevo paciente o una nueva bioseñal, de no tenerlos la sección de nuevo ingreso no será visible.
- Cuando se genera un nuevo usuario, el archivo que se sube es renombrado con el nombre del paciente con el fin de que en la sección de búsqueda se puedan relacionar bien
- Cuando se desea ingresar un paquete de señales, el usuario debe renombrarlo con el mismo identificador del paciente con el fin de que sea asociado a él cuando se realice una búsqueda, en caso de no estar relacionado con ningún paciente, se podrá subir y consultar desde la sección de búsqueda en la pestaña de señales

Una vez el paciente se ha ingresado satisfactoriamente, se puede ir a la sección de búsqueda, en esta sección se encontrarán dos pestañas, en una de ellas es posible buscar pacientes, inicialmente se cargan todos los pacientes existentes en la base de datos, sin embargo, también es posible realizar la búsqueda de un paciente en específico a través de su nombre o identificador único. En la figura 16 se muestra la interfaz de búsqueda de paciente:

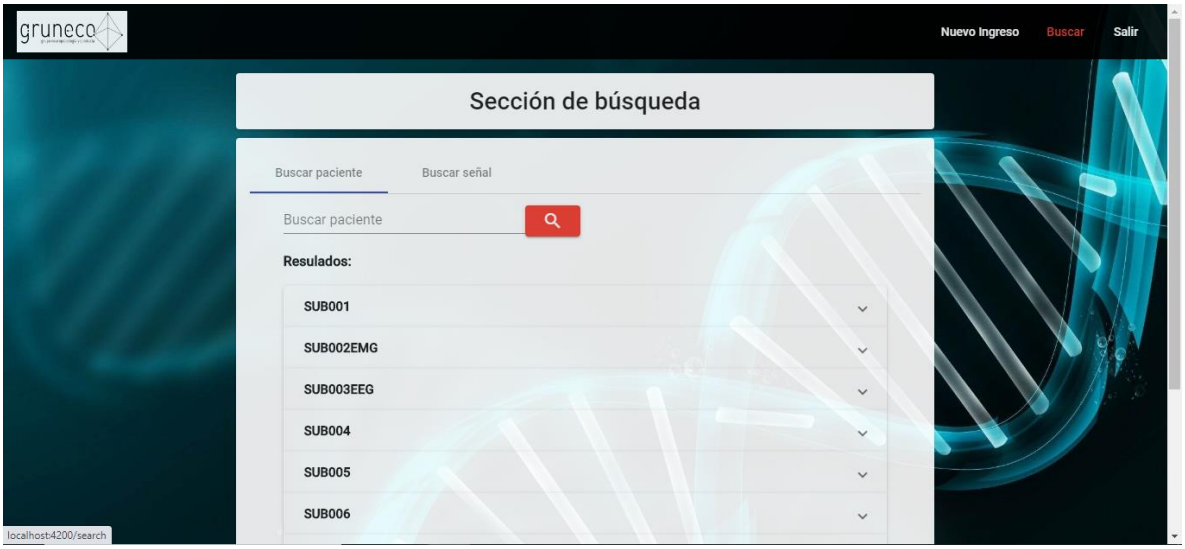

Figura 16. Sección de búsqueda de pacientes

Para esta pestaña se considera lo siguiente:

- Se puede buscar un paciente poniendo su identificador exacto, en caso de no tenerlo, se puede realizar la búsqueda utilizando secciones del nombre y el sistema arrojará los nombres que coincidan con la búsqueda

- Al desplegar el recuadro de un paciente, se podrá revisar toda su información
- En la sección de archivos, se vizualizarán todos los archivos alojados en la nube que coincidan con el identificador del paciente
- Debido a que cada vez que se despliega un paciente se realiza una búsqueda en tiempo real de los archivos asociados a él, solo se mostrarán los archivos que estén disponibles para su descarga, por lo que si se da clic en cualquiera de ellos, iniciará automáticamente la descarga
- Se cuenta con un botón de editar con el cual se podrá modificar la información básica del paciente o agregar nuevas observaciones según sea el caso
- Adicionalmente se cuenta con un botón para eliminar el paciente, al dar clic en él, se eliminará automáticamente de la base de datos
- Las opciones de visualización, edición y borrado, dependen de los permisos que tenga el usuario dentro de la plataforma

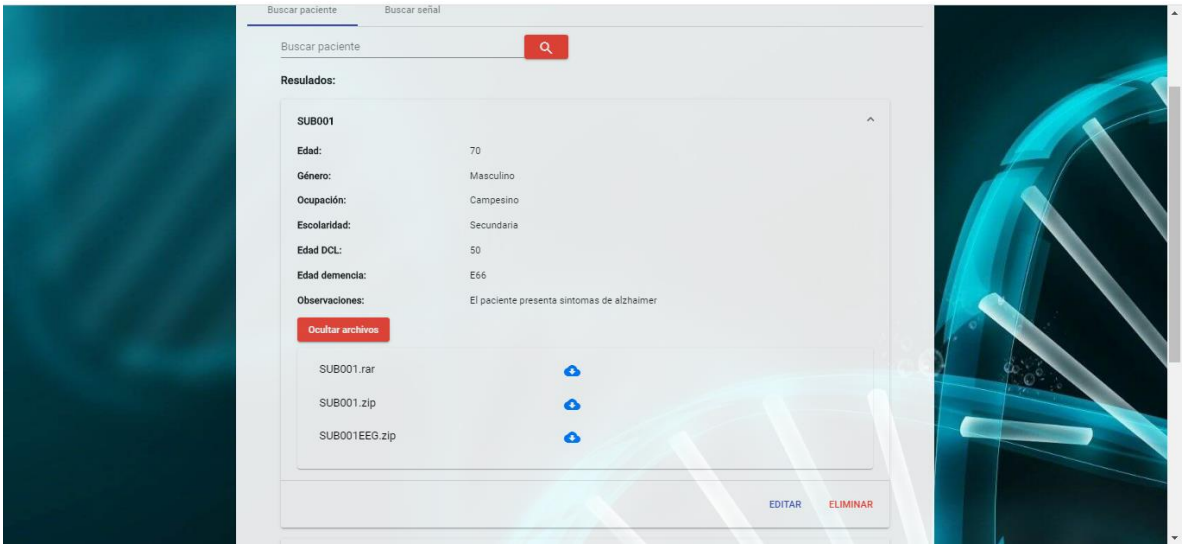

Figura 17. Despliegue de información de un paciente

Adicionalmente se cuenta con una pestaña para buscar únicamente señales, del mismo modo que con la pestaña pacientes, al dar clic sobre esta pestaña se muestran todas las señales almacenadas y también es posible buscar por el nombre exacto del archivo o también buscará coincidencias con lo que se escriba en el campo de búsqueda. En la figura 18 se visualiza la interfaz de la pestaña de búsqueda de bioseñal:

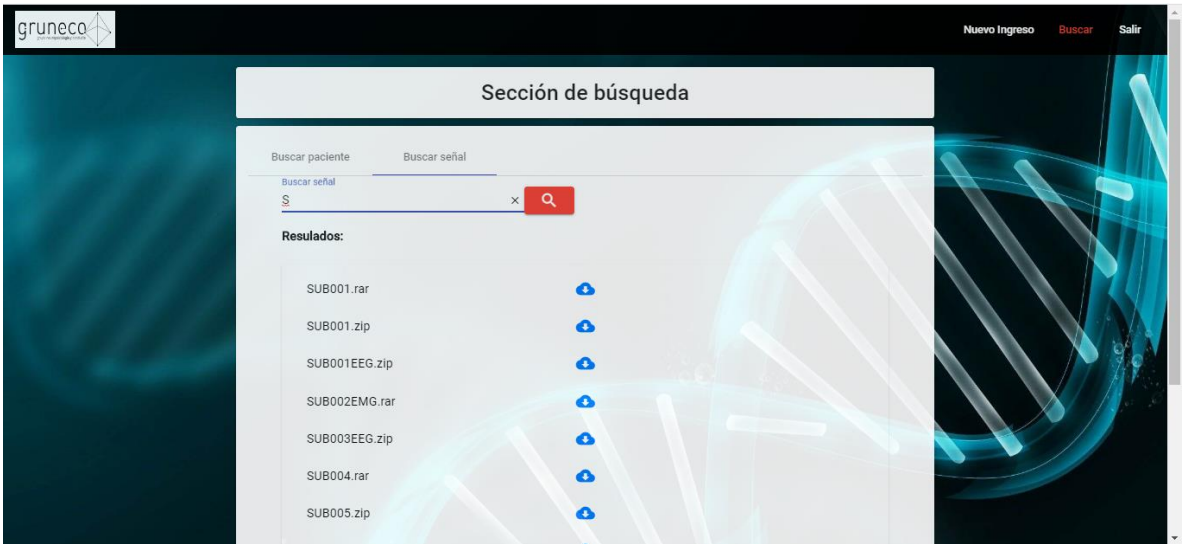

Figura 18. Pestaña de búsqueda de bioseñal

En el momento en que el usuario desee salir de la plataforma, únicamente debe dar clic en el botón "Salir" ubicado en la esquina superior derecha, al dar clic sobre este botón, será redireccionado a la página de autenticación e internamente el token con el que se manejaba su sesión, será destruído con el fin de nadie más pueda ingresar con los datos de su sesión.

6.3. Esquema de carpetas y detalles principales del proyecto

El proyecto se dividió en dos carpetas o secciones principales, en una de ellas, se alojó toda la lógica perteneciente al backend tales cómo middlewares, APIs, conexiones y operaciones con base de datos e interacciones con AWS. En la carpeta o sección restante, se desarrolló todo lo relacionado con el frontend como las interfaces, configuraciones, lógica para el consumo de APIs, entre otros.

Para la sección del backend, se cuenta con el siguiente esquema:

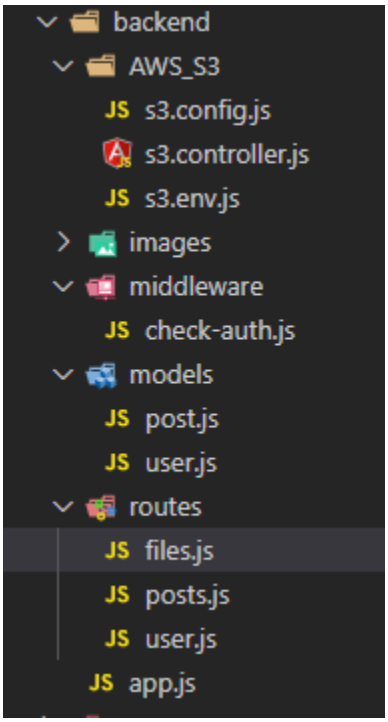

Figura 19. Esquema de carpetas del backend

Esta carpeta está dividida en archivos y subcarpetas las cuales se detallan a continuación:

- **AWS_S3:** En esta carpeta se encuentran todos los controladores necesarios para acceder al servicio de almacenamiento en la nube S3. En el archivo "*s3.env.js"* se almacenan las credenciales necesarias para acceder remotamente al servicio de almacenamiento, estas claves son generadas a partir de la configuración inicial de AWS. El archivo "*s3.config.js*" es el encargado de realizar la conexión con los servicios de AWS. Finalmente el archivo "*s3.controller.js*" contiene toda la lógica para realizar la consulta de los archivos, la carga y la descarga
- **images:** En esta carpeta se aloja temporalmente el archivo cargado al ingresar un nuevo paciente, en el momento en que se carga el archivo completo en el servidor, inicia la carga del archivo a AWS y tras completarse dicha carga, el archivo es eliminado
- **middleware:** En esta carpeta se deben alojar todas las comprobaciones necesarias para realizar determinada acción, en el momento se cuenta con el archivo "*check-auth.js*" el cual verifica que el usuario que realizó cierta petición, se encuentre correctamente autenticado y con una sesión válida

- **models:** En esta carpeta se definen los modelos de los objetos que serán utilizados por parte del backend, para este proyecto se cuenta con el modelo "*post.js*" que corresponde a los usuarios ingresados con su respectiva información y el modelo "*users.js*" que corresponde a los usuarios registrados en la plataforma
- **routes:** Esta carpeta almacena la lógica de cada una de las APIs desarrolladas para manejar los archivos, los pacientes y los usuarios.
- **Archivo *"app.js"*:** En este archivo se establecen las características principales de la aplicación, tales cómo la conexión con la base de datos remota, configuración de headers para el control CORS (Cross-origin resource sharing, en sus siglas en inglés) y en ella también se alojan las rutas para el consumo de las APIs

Del lado del frontend, se tiene el siguiente esquema de carpetas y archivos:

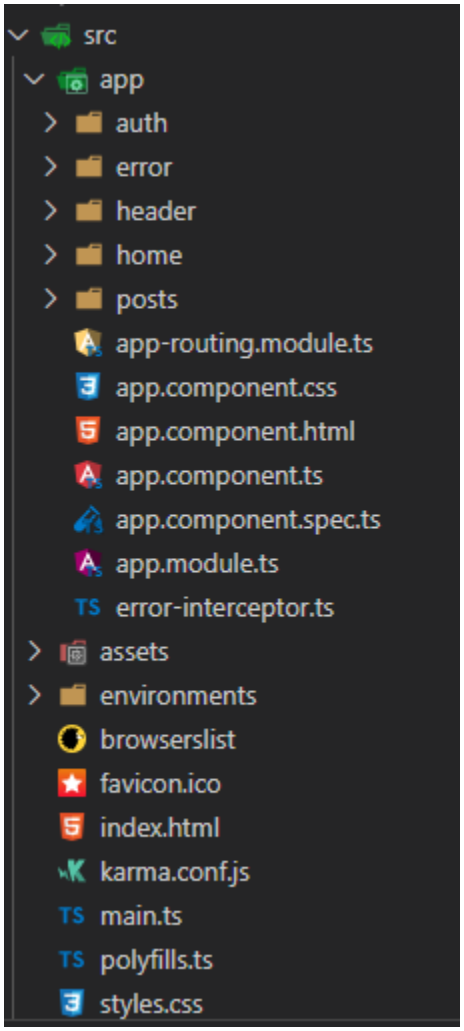

Figura 20. Esquema de carpetas y archivos para frontend.

Esta sección cuenta con subcarpetas, archivos relevantes desarrollados para el proyecto y algunos archivos que no serán mencionados ya que hacen parte de la configuración de Angular y son generados automáticamente cuando se genera el nuevo proyecto. A continuación el detalle de los archivos y carpetas relevantes:

- **auth:** En esta carpeta se encuentran dos subcarpetas relacionadas con la interfaz de registro y con la interfaz de ingreso, en ellas se puede encontrar el template de cada una de ellas, las hojas de estilo y la lógica que hace la conexión con el backend y hace el consumo de las APIs. Adicionalmente, se encuentran algunos servicios que ayudan a comprobar que existe una sesión válida y que de no ser así, redirige al usuario a la interfaz de ingreso. El detalle de estos componentes se puede apreciar en la figura 21:

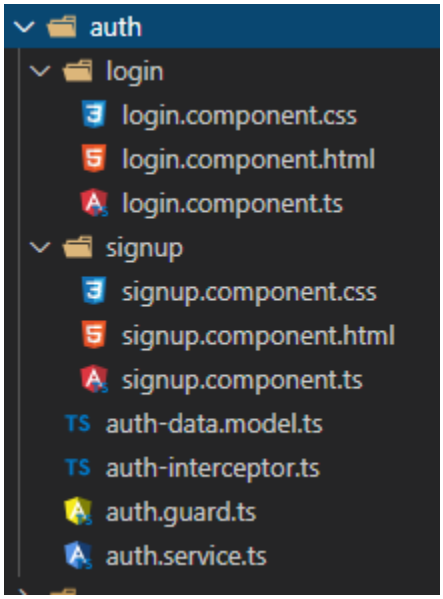

Figura 21. Detalles del componente auth.

- **error:** En esta carpeta se encuentra el template para el componente del mensaje de error, este componente solo es utilizado cuando el servidor responde con un status de error y es inyectado por el servicio definido en el archivo "*error-interceptor.ts*"
- **header:** En este directorio se encuentra el componente que se puede observar en la parte superior de cada interfaz. Ya que Angular tiene una estructura definida por componentes, cada uno de ellos puede ser usado en la interfaz que se requiera conservando todos sus estilos

- **home:** En esta carpeta se aloja la interfaz de bienvenida en donde se muestra información relevante y detalles de contacto
- **posts:** Aquí se encuentra todo lo relacionado con el ingreso, busqueda y visualización de pacientes y bioseñales, desde los templates y hojas de estilos, hasta los servicios y la lógica requerida para el consumo de las APIs. Cada función está separada en subcarpetas que pueden ser detalladas en la figura 22:

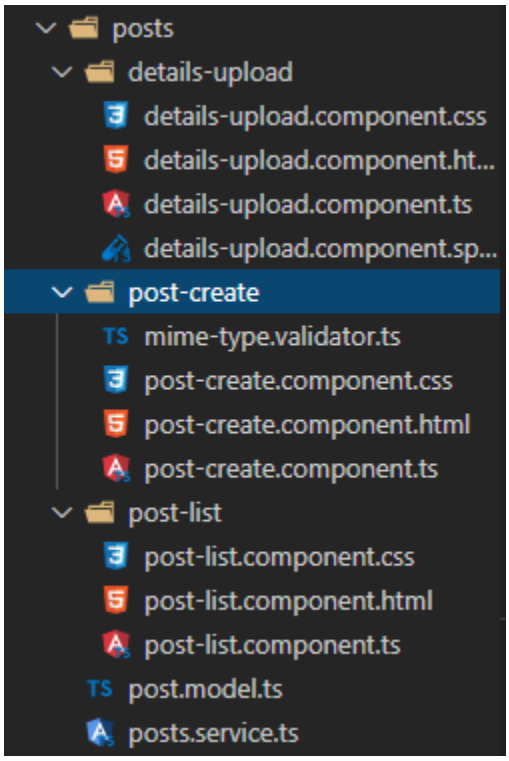

Figura 22. Detalle de la carpeta posts

- **Archivo "*app-routing.module.ts*":** En este archivo se relaciona cada una de las interfaces mencionadas anteriormente con una ruta específica, de tal modo que se pueda realizar correctamente la navegación dentro de la plataforma
- **Archivo "*app.module.ts*":** Es quizá uno de los archivos más importantes en el proyecto ya que en él se importan todas las librerías requeridas para el proyecto y además allí se deben preestablecer todos los componentes creados para la aplicación, de lo contrario no será posible usar el componente ni será posible relacionarlo con ninguna ruta
- **Archivo "*index.html*":** Este archivo es el principal template de la aplicación y sobre el cual se define el estilizado principal de la aplicación y además se establece la etiqueta "<*app-root*>" lo cual

significa que sobre él se renderizaran todos los componentes definidos en la aplicación web

Finalmente se cuenta con un archivo en común entre el backend y el frontend llamado "*package.json*" en este archivo se definen las configuraciones de inicio del proyecto en un ambiente local y además se listan todas las librerías y dependencias usadas en el proyecto.

6.4. Guía de instalación del proyecto para modificaciones en un ambiente local

1. Descargar el instalador de Node.JS desde la página: https://nodejs.org/es/.
2. Una vez instalado, se abre una terminal y se navega hasta la carpeta que contiene el proyecto y se ejecuta el comando "*npm install*". Este comando revisa y descarga todas las librerías y dependencias listadas en el archivo "*package.json*" incluidas las del framework Angular.
3. Cuando se finalice la descarga e instalación de las librerías y dependencias se debe ingresar el siguiente comando en la terminal "*npm run start:server*", este comando habilitará un servidor local para el consumo de servicios y esta preconfigurado para que se ejecute en el puerto 3000.
4. Sin cerrar la terminal en la que se está ejecutando el servidor local, se debe abrir otra terminal y navegar nuevamente hasta la carpeta que contiene el proyecto y ejecutar el comando "*ng serve*", este comando cargará todos los elementos y las interfaces del frontend y será posible acceder a él a través del navegador ingresando la ruta http://localhost:4200/.

6.5. Guía para el despliegue de la aplicación utilizando AWS

1. Ingresar a la cuenta de AWS y en la pestaña de servicios, seleccionar la opción "*Elastic Beanstalk*", esta herramienta permitirá alojar el backend en la nube de Amazon.

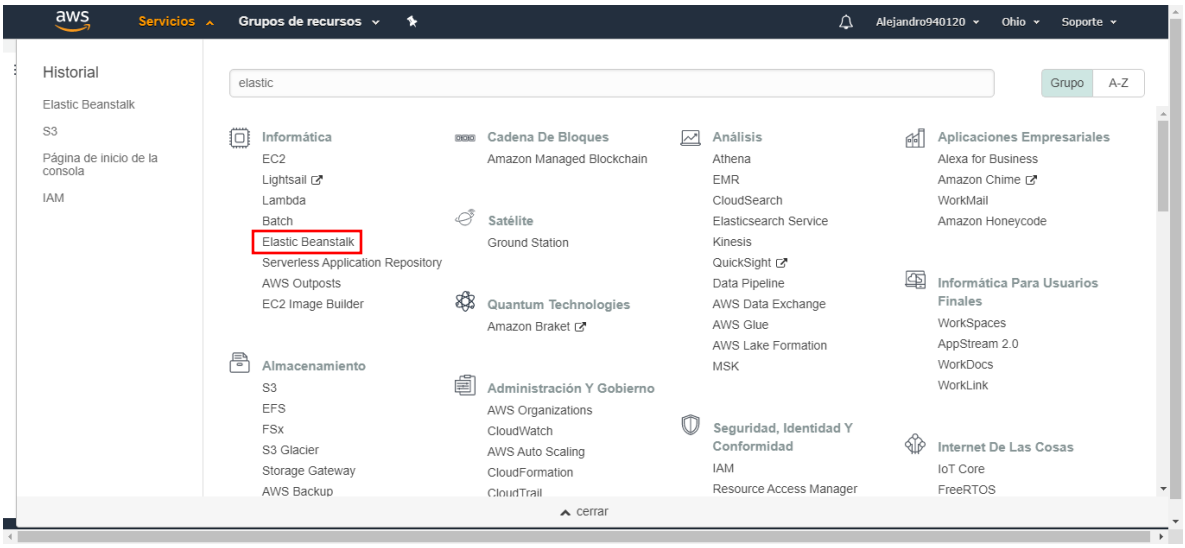

Figura 23. Consola de AWS

2. Una vez se cargue la interfaz de la aplicación, se debe dar clic en la opción "*Create Application*" para proceder a configurar el entorno.

Figura 24. Interfaz AWS Elastic Beanstalk

3. Después de dar clic sobre el botón anterior, aparecerá un menú en donde se debe ingresar el nombre de la aplicación, el lenguaje con el que correrá la aplicación, esto se muestra en las figuras 25 y 26.

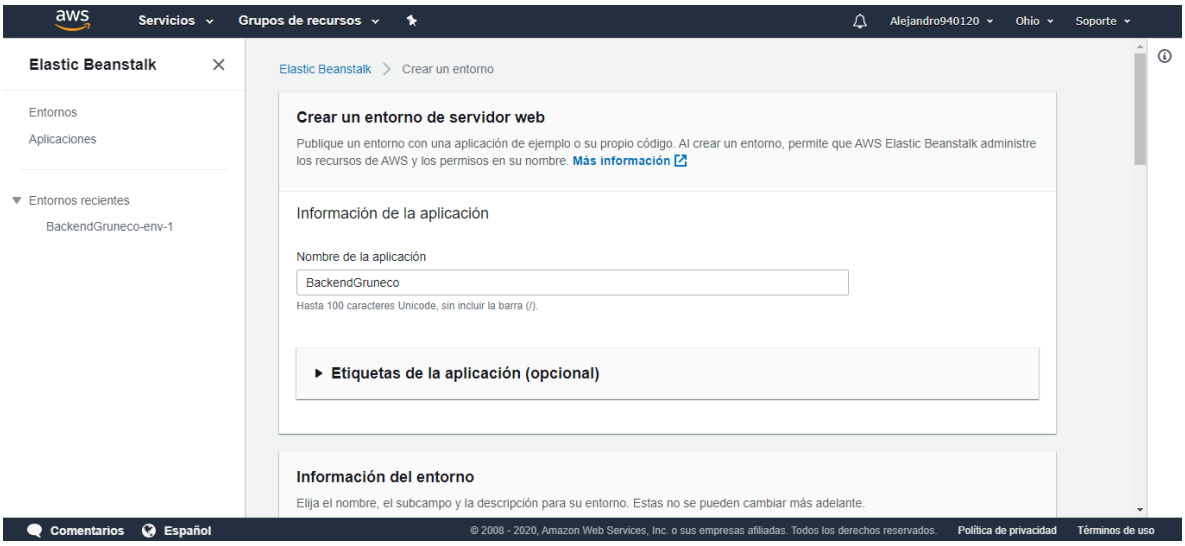

Figura 25. Menú de configuración de entorno web, nombre

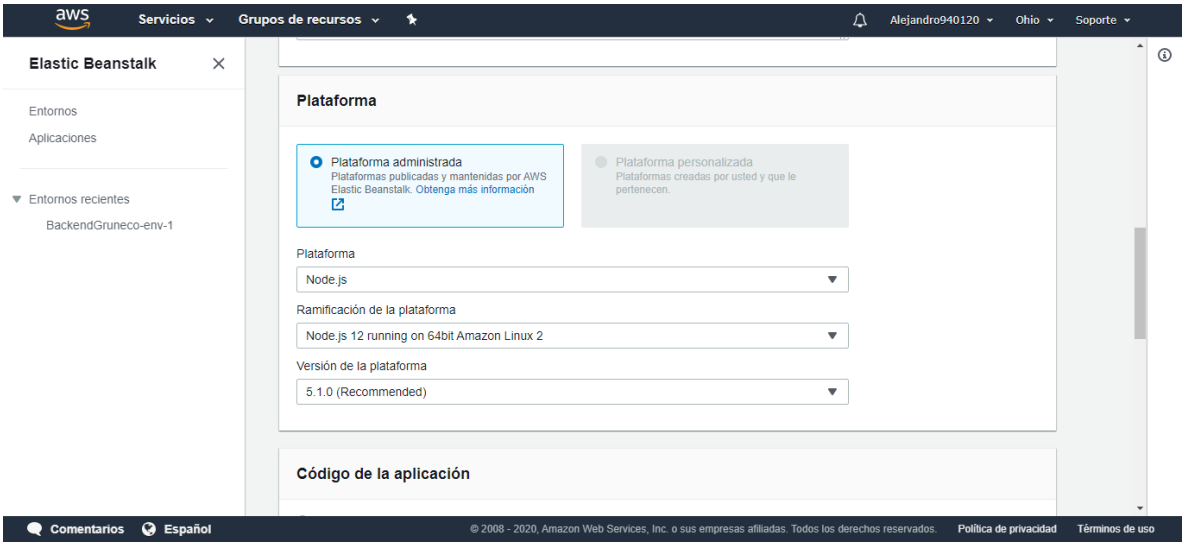

Figura 26. Menú de configuración de entorno web, lenguaje

4. Para la sección final de la configuración del entorno, es necesario dirigirse a la carpeta backend del directorio del proyecto principal y comprimir todos los elementos es un archivo .zip, es importante que tenga este tipo de compresión ya que de lo contrario podría presentar incompatibilidades. Una vez se realice lo anterior, se debe volver a la consola de configuración de *Elastic Beanstalk* y se selecciona el botón "**elegir archivo**" señalado en la figura 28 y se selecciona el archivo previamente comprimido y da clic en el botón "*Crear un entorno*".

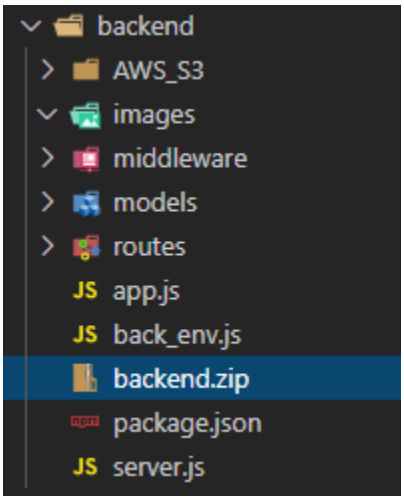

Figura 27. Archivo de Backend comprimido.

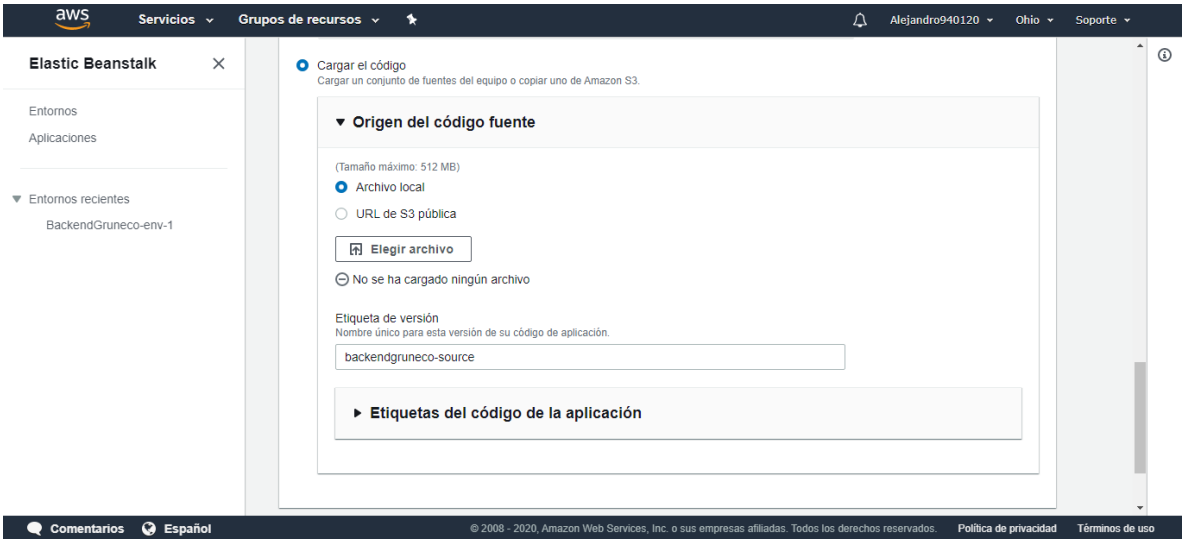

Figura 28. Selección de archivo comprimido

5. Luego de dar clic en el botón, se empezarán a instalar las dependencias de la aplicación especificadas en el archivo "*package.json*" y la consola dará reportes en tiempo real sobre el estado de la instalación tal y como se muestra en la figura 29.

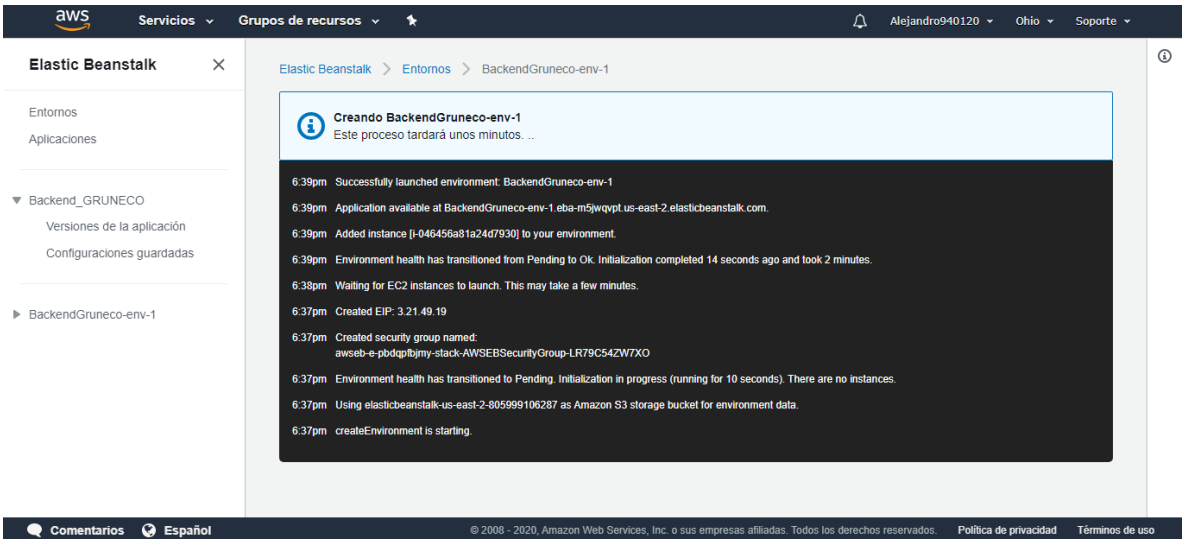

Figura 29. Estado de instalación del entorno web

6. Cuando la interfaz de la consola se recargue, el entorno se habrá generado correctamente, sin embargo, aún no será posible usarlo por la aplicación ya que este entorno genera una IP que debe ser agregada como una excepción dentro del cluster de base de datos. Para esto se debe copiar la dirección EIP señalada en la figura 30.

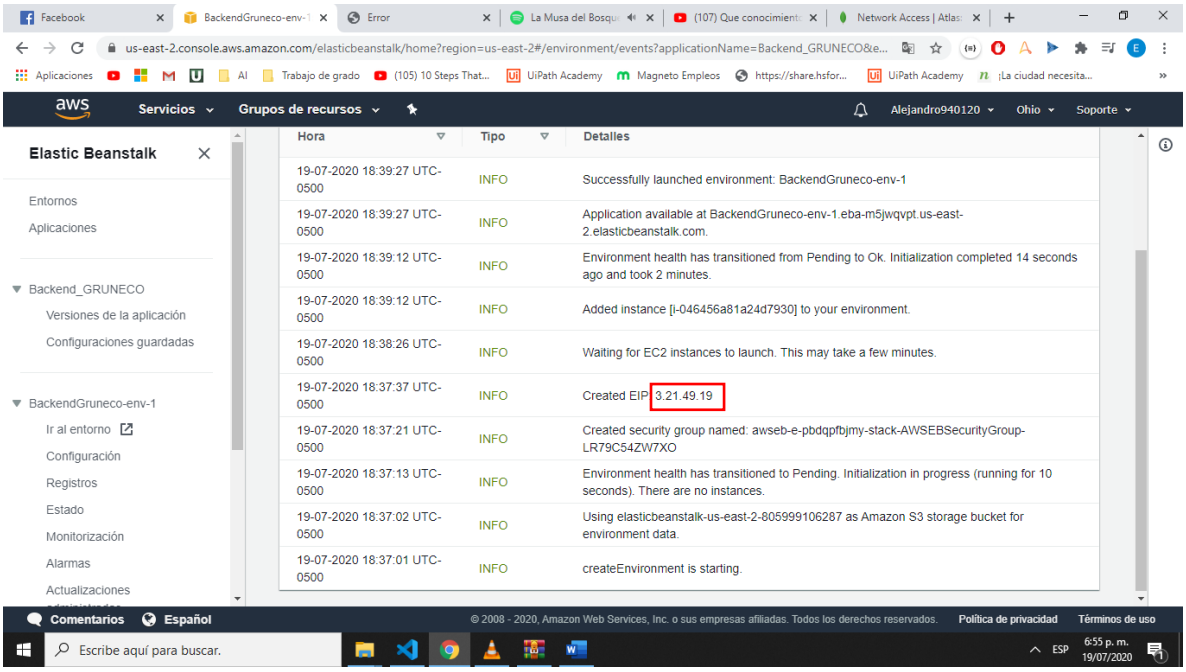

Figura 30. EIP del entorno web en Elastic Beanstalk

Una vez se obtenga esta EIP, se debe ingresar al cluster creado en la plataforma MongoDB Atlas (Para la creación del cluster se recomienda seguir el siguiente video:

https://www.youtube.com/watch?v=TLPtf9tw7EY), seleccionar el cluster que aloja la base de datos de la aplicación, dar clic en la opción "*Network Access*" de la pestaña "*SECURITY*", dar clic sobre el botón "*+ ADD IP ADDRESS*", ingresar la EIP generada anteriormente y esperar a que su estado quede en "*Active*".

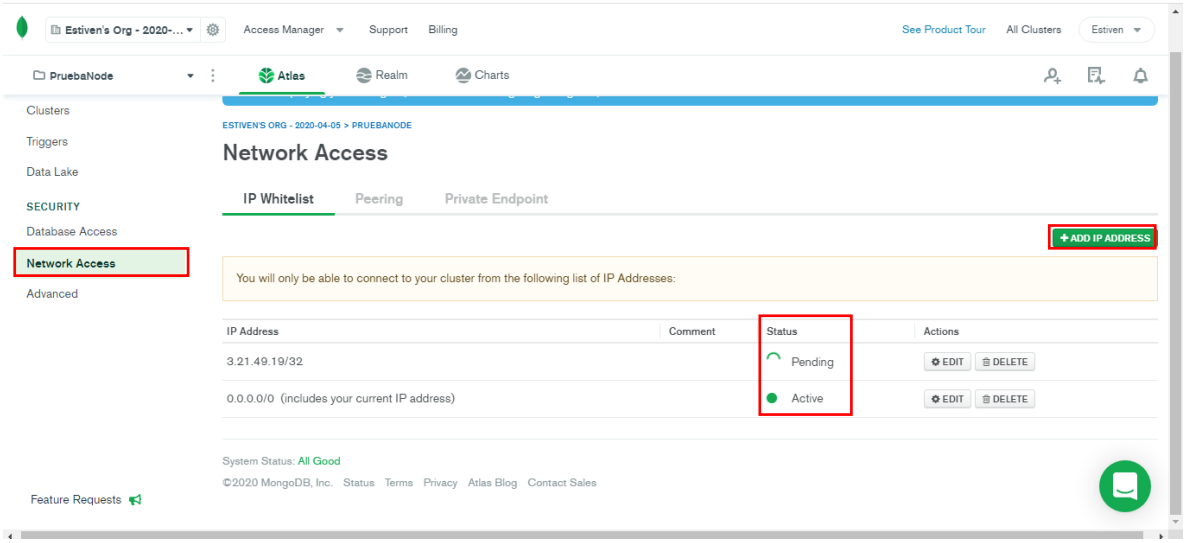

Figura 31. Network Access de MongoDB Atlas

7. Una vez se ingrese la EIP, será necesario reiniciar el servidor de la aplicación para que pueda establecer la conexión con la base de datos, una vez se reinicie, el backend de la aplicación quedará totalmente funcional y su dirección será la que se indica en la figura 32:

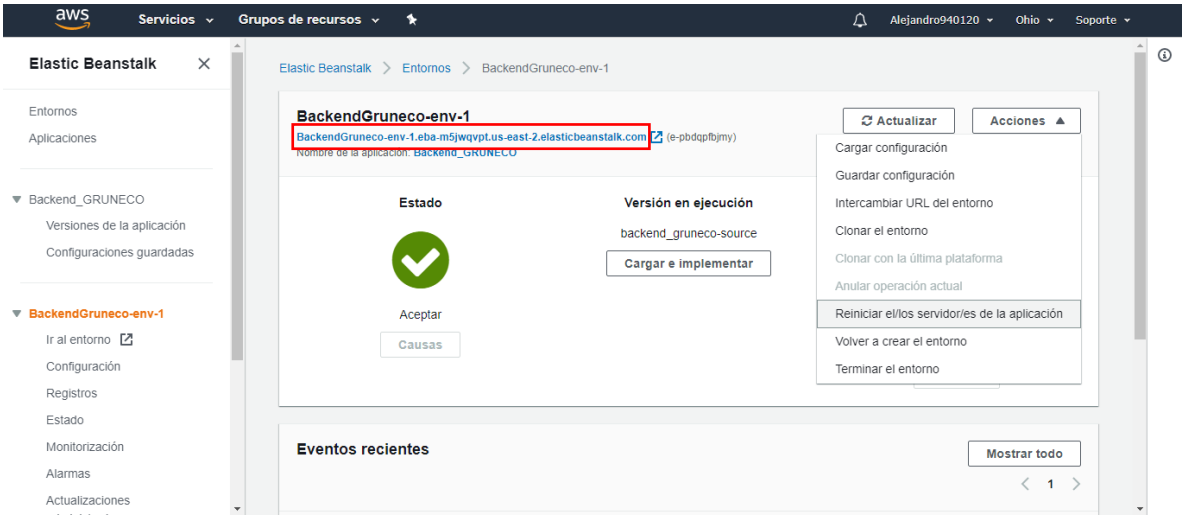

Figura 32. Dirección del servidor backend

8. Una vez configurado el backend de la aplicación, se debe configurar el frontend. El primer paso es copiar la dirección mostrada en la figura 32 y pegarla en el archivo "*environment.prod.ts*" que se encuentra en la carpeta "*environments*" del directorio principal de la aplicación. Una vez realizado esto, se debe abrir una consola y escribir el comando "*ng build –prod*", esto generará los componentes necesarios para realizar la compilación del frontend de la aplicación en un ambiente productivo.

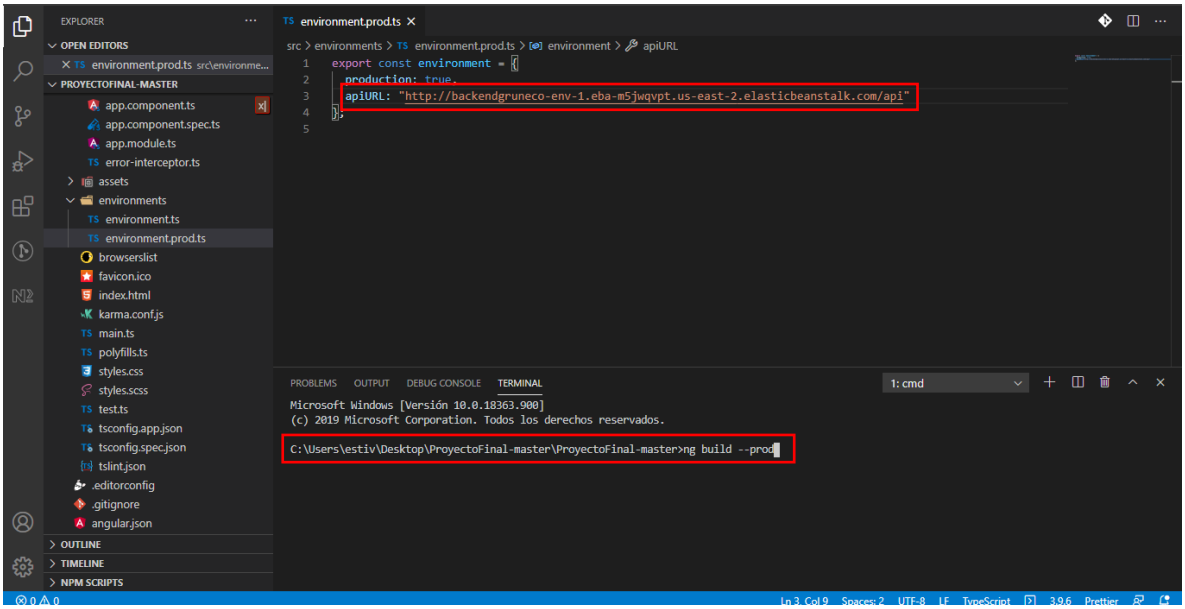

Figura 33. Configuración para versión de producción de la aplicación

La versión de producción de la aplicación se podrá encontrar en la carpeta "*dist*" del directorio principal de la aplicación y deberá contener los archivos mostrados en la figura 34.

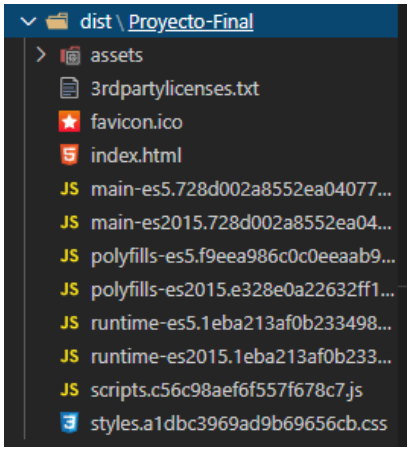

Figura 34. Archivos de producción de la aplicación

9. El siguiente paso consiste en ingresar nuevamente a la consola de AWS, y seleccionar la opción S3.

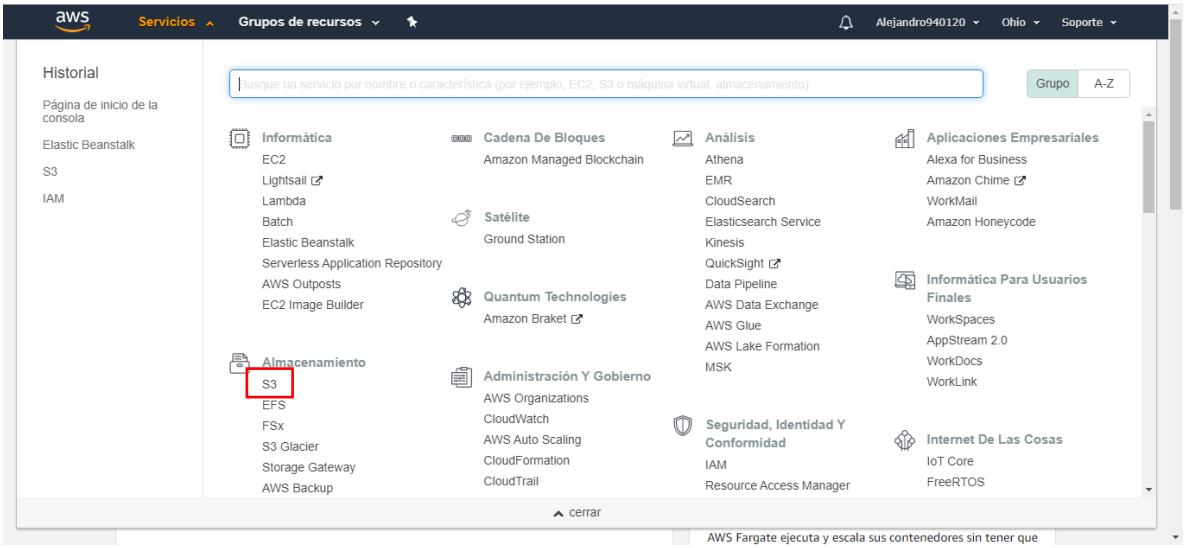

Figura 35. Consola de AWS

10. Una vez se cargue la interfaz de S3, se debe crear un nuevo bucket, ingresar a él y cargar los archivos alojados en la carpeta "dist" descritos en el paso 8. A continuación se muestra el paso a paso para esta sección:

    - Clic en botón *"+ Crear bucket"*

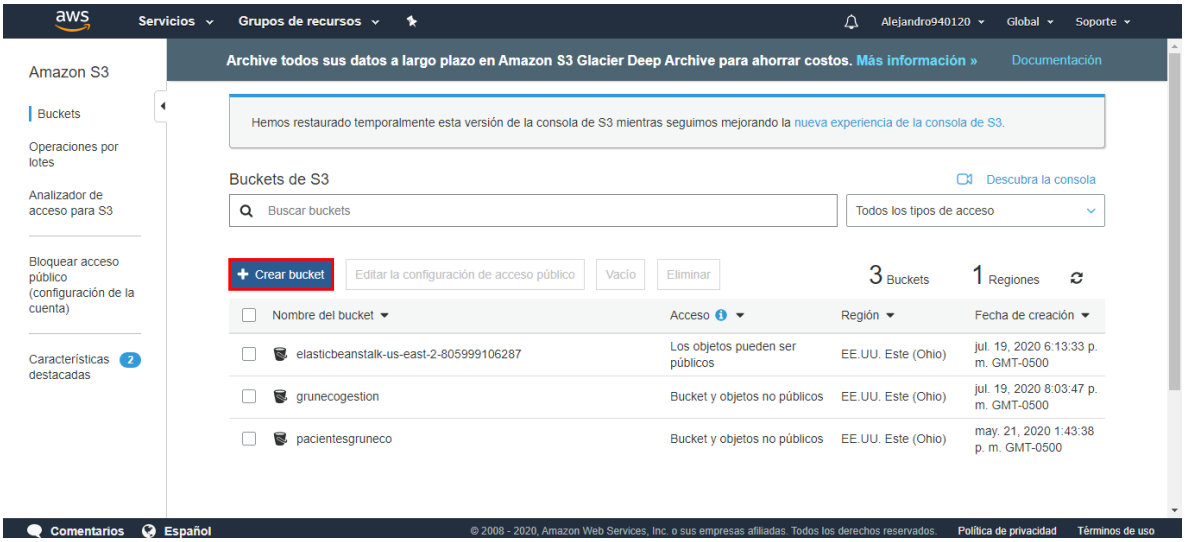

Figura 36. Creación de bucket, paso 1

- Ingresar nombre y dar clic en el botón crear

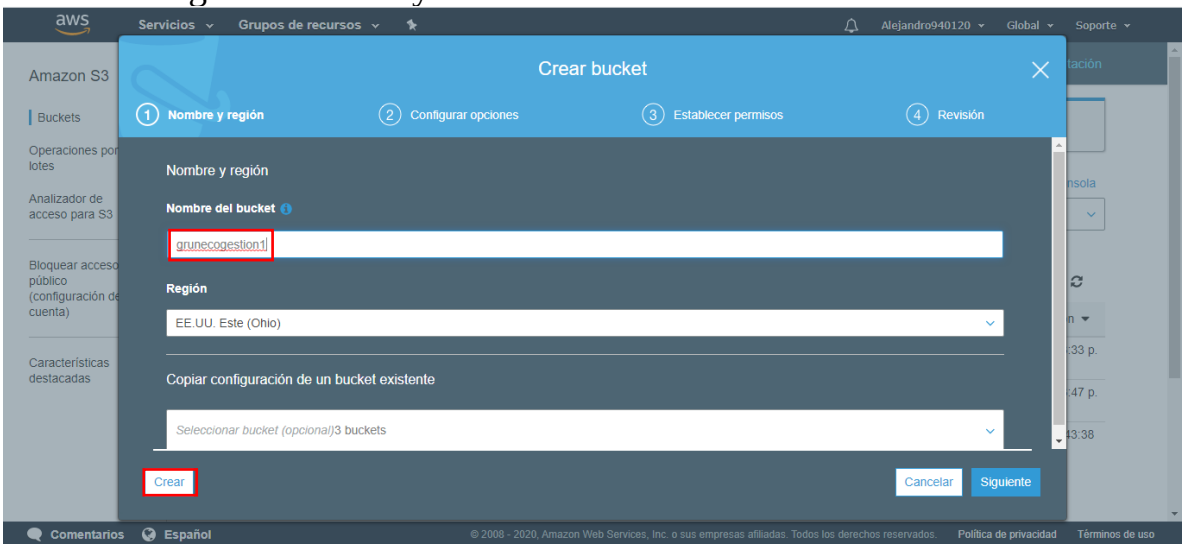

Figura 37. Creación de bucket, paso 2

- Dar clic en el botón cargar

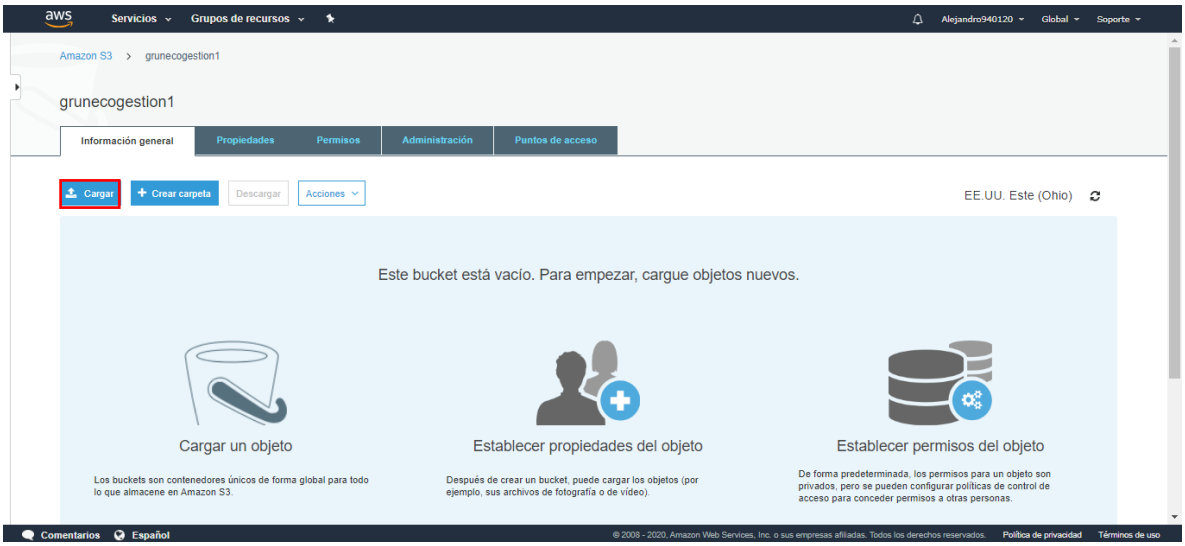

Figura 38. Carga de archivos de producción, paso 1

- Seleccionar los archivos de producción y dar clic en el botón “abrir”

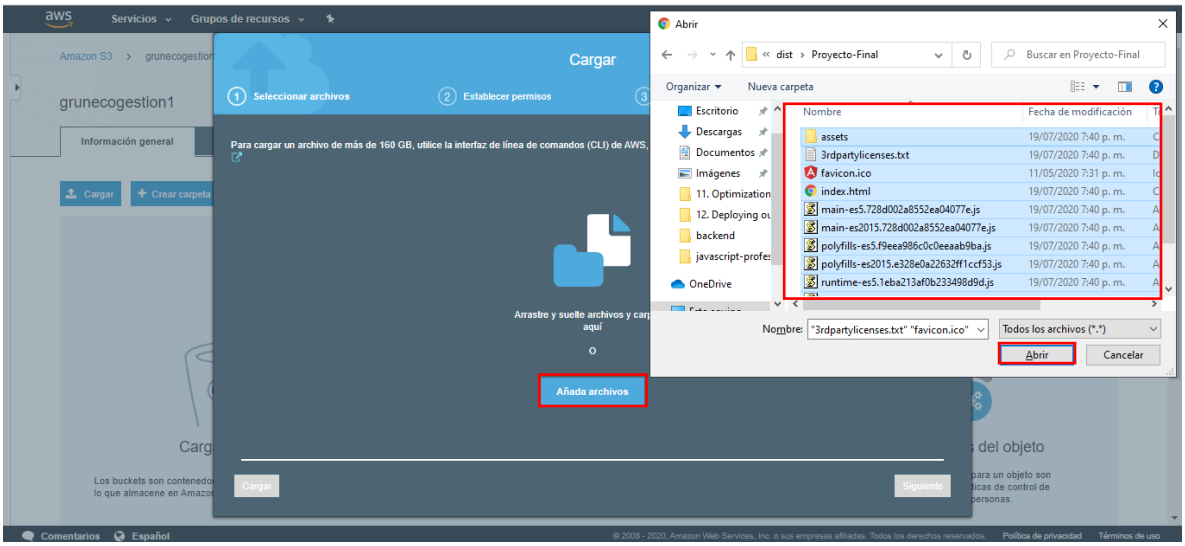

Figura 39. Carga de archivos de producción, paso 2

- Finalmente se da clic en el botón cargar y ya los archivos de producción quedarán almacenados en el bucket, se debe tener cuidado con las carpetas ya que en ocasiones no se cargan correctamente y es necesario hacer la carga de estas por separado

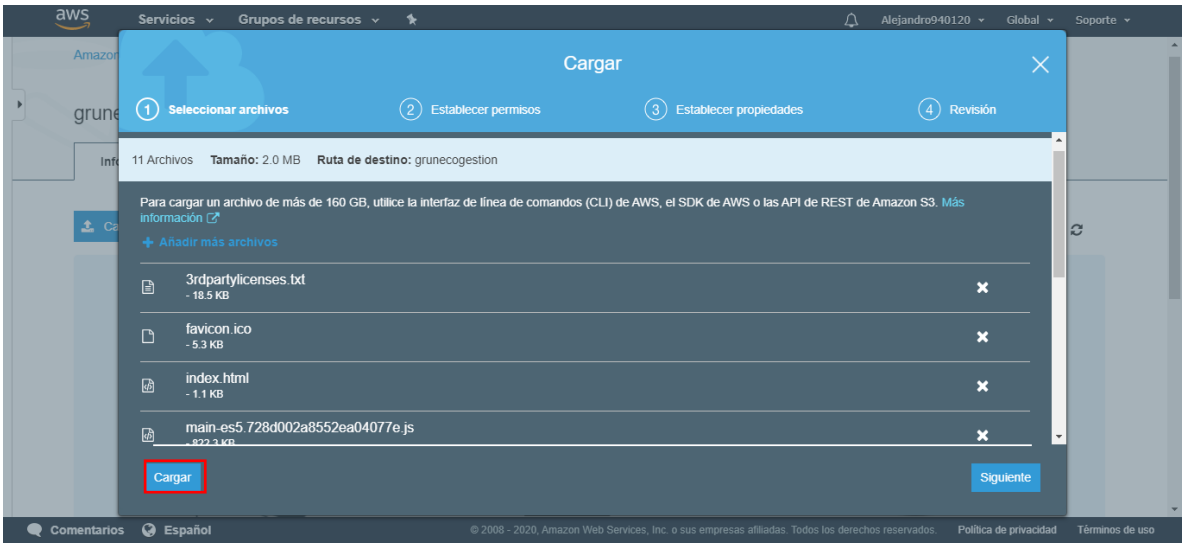

Figura 40. Carga de archivos de producción, paso 3

11. Ahora es necesario configurar el bucket para que sea de acceso público de solo lectura, esto permitirá que cualquier usuario pueda ingresar a la aplicación web. Para esto se debe ingresar en la pestaña *"Permisos"* y seleccionar la opción *"Política del bucket"*.

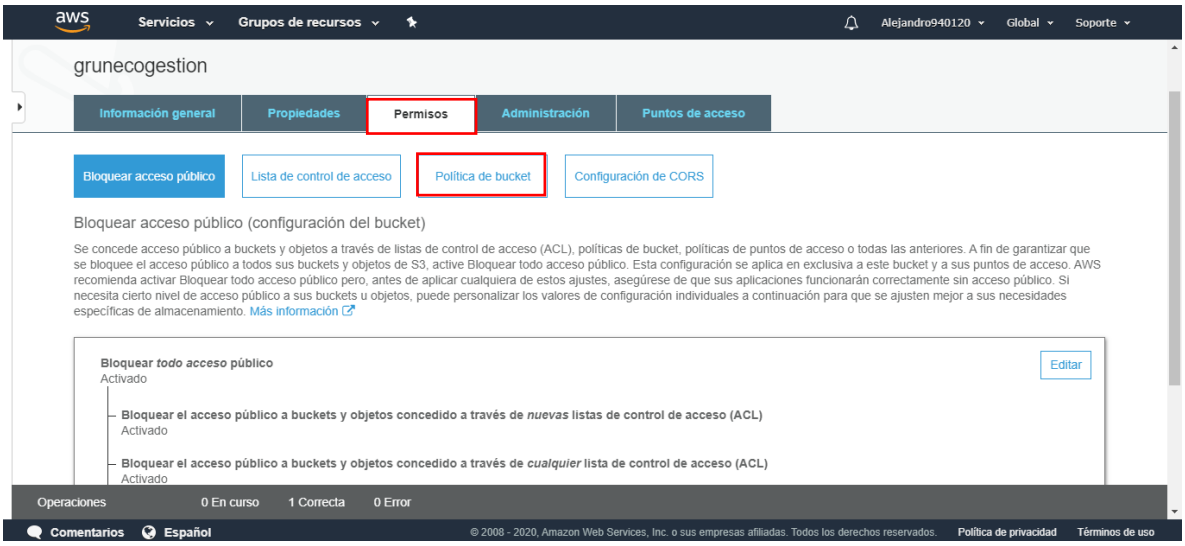

Figura 41. Configuración de permisos

Se deben ingresar las siguientes instrucciones en el recuadro de comandos:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicRead",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::grunecogestion/*"
    }
  ]
}
```

Estas instrucciones habilitarán el acceso público al bucket lo que permitirá configurarlo como un hosting para aplicaciones web.

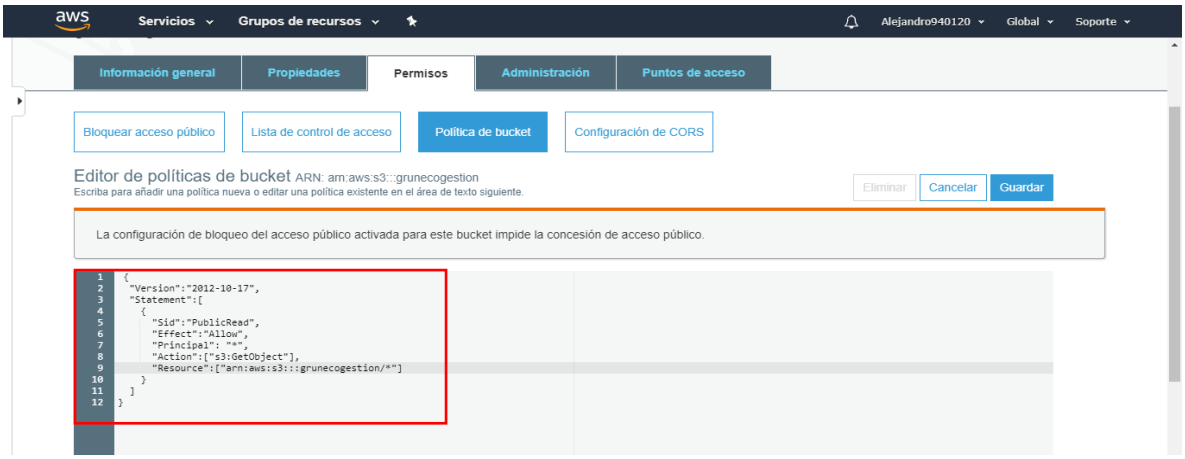

Figura 42. Configuración de privacidad del bucket

12. Una vez configurada la política de privacidad, se debe seleccionar la pestaña *"Propiedades"* y desplegar la opción *"Alojamiento de sitios web estáticos"*.

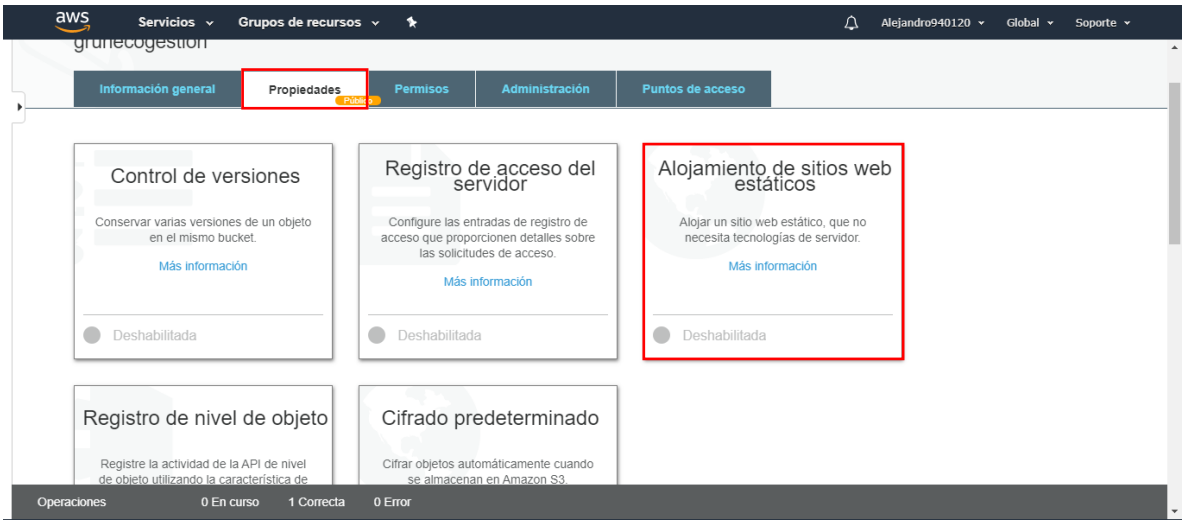

Figura 43. Configuración del bucket para ambiente web

13. Finalmente se debe seleccionar la opción *"Usar este bucket para alojar un sitio web"* e ingresar *"index.html"* en los campos *"Documento de índice"* y *"Documento de error"* y al dar en el botón guardar, ya quedará habilitado el sitio web que será accesible desde el enlace mostrado en la figura 43.

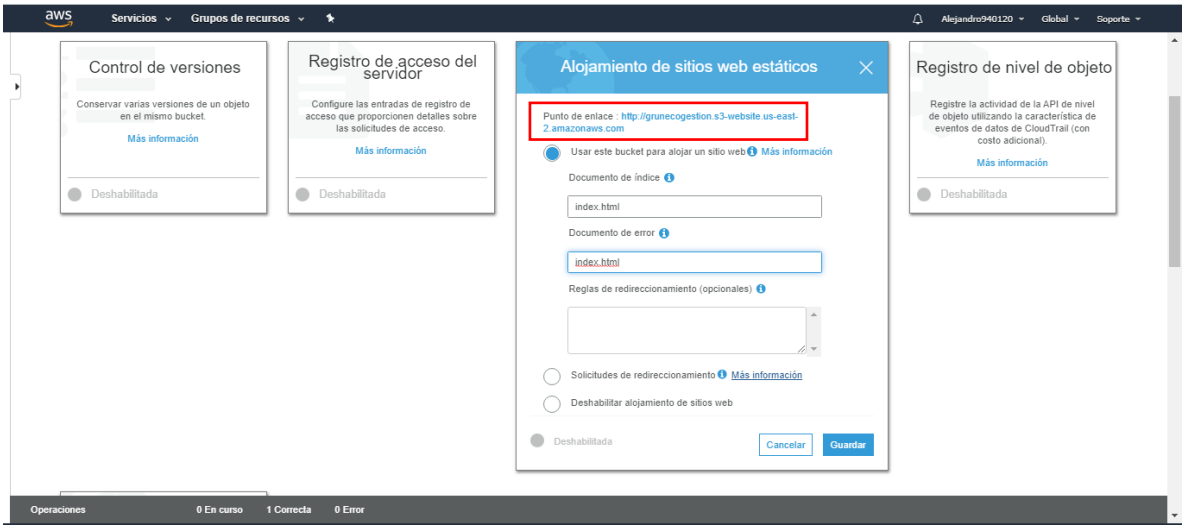

Figura 43. Configuración y enlace de acceso al sitioweb

Una vez desplegada la plataforma, se procedió a realizar pruebas de funcionamiento en producción y se logró tener una funcionalidad igual a la que se tenía en un entorno local ya que se realizó un flujo similar al mostrado en la sección 6.2 y todas las funcionalidades desarrolladas se encontraban disponibles y al ser usadas no generaban ningún error ni de lógica ni de servidor, además, la aplicación también contó con un buen rendimiento gracias a la infraestructura que proporcionan los servicios en la nube de AWS.

La aplicación fue probada en un computador de escritorio (figura 44) y aunque no está optimizada para ambientes móviles, también se probó su funcionamiento en un celular (figura 45), en ambos ambientes la aplicación funcionó sin problema, sin embargo es necesario activar el modo *"Escritorio"* en el celular para que se pueda vizualizar correctamente.

Estas pruebas demuestran que la aplicación después de ser desplegada en los servidores de AWS, puede ser accedida desde cualquier dispositivo con acceso a internet desde cualquier lugar del mundo y debido a esto se podría optimizar el intercambio de información entre integrantes de un grupo de investigación ya que al contarse con un esquema centralizado de información, es posible acceder a ella de una forma más fácil, estructurada y rápida.

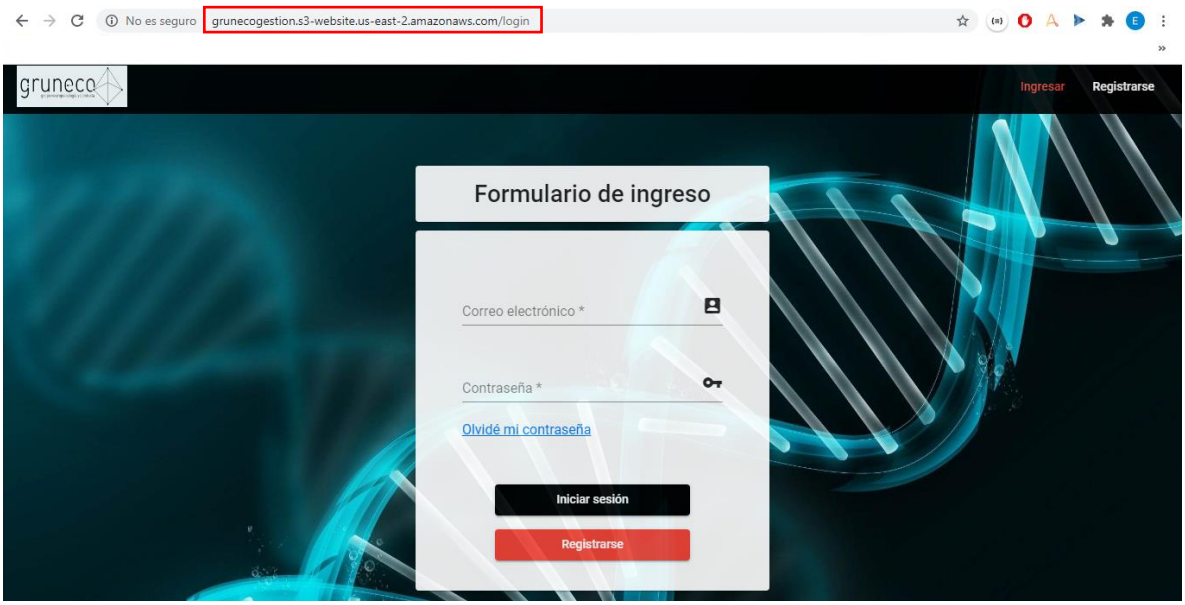

Figura 44. Visualización de la aplicación en un computador de escritorio

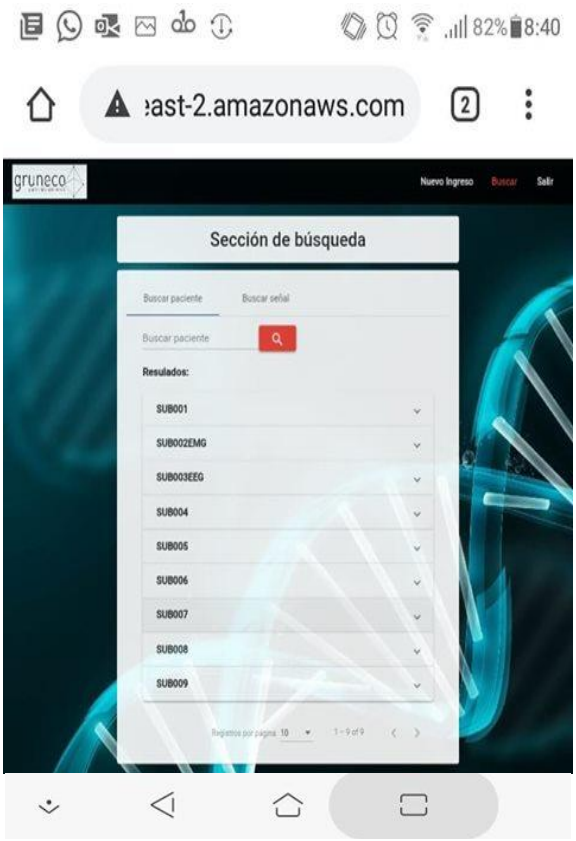

Figura 45. Visualización de la aplicación en un ambiente móvil

7. CONCLUSIONES

- En una era en la cual los datos son cada vez más importantes, se convierte en una necesidad estandarizar la forma en la que estos son almacenados y administrados, para lograr esto, el uso de las herramientas web se convierte en una de las mejores opciones ya que ofrecen la posibilidad de lograr una centralización de los datos y a su vez garantizar una disponibilidad permanente de estos desde cualquier dispositivo e inclusive desde cualquier lugar del mundo con acceso a internet.
- Ya que en ocasiones es necesario trabajar con datos que pueden ser sensibles o confidenciales, es necesario blindar la aplicación tanto desde el frontend como en el backend para evitar el acceso de usuarios no deseados, para esto, es necesario hacer un exhaustivo control de pruebas con el fin de detectar vulnerabilidades e implementar soluciones eficientes que complementen las que vienen por defecto con el stack de herramientas que se está usando. Para este proyecto fue posible implementar soluciones como encriptación de alto grado para las contraseñas de los usuarios, niveles de permisos de usuarios, JWT y adicionalmente se implementaron un AuthGuard el cual se encarga de realizar un redireccionamiento a cualquier ingreso irregular a la plataforma.
- La computación en la nube se ha convertido en una de las tecnologías más importantes de los últimos tiempos ya que ofrece capacidades prácticamente ilimitadas, desde el almacenamiento de archivos, bases de datos y despliegue de aplicaciones como se realizó en este proyecto, hasta procesamiento de grandes volúmenes de datos en tiempo récord, debido a esto se hace cada más necesario migrar de los sistemas de información tradicionales a sistemas que estén alojados en la nube ya que estos ofrecen una mayor variedad de herramientas y adicionalmente cuentan con la infraestructura para garantizar que las aplicaciones o plataformas alojadas en esta, siempre estén disponibles para su uso inmediato.
- Con el fin de abordar una problemática latente la cual consiste en la falta de información de bioseñales de la población nacional, a través de esta plataforma se podrá crear un repositorio con información centralizada de la población nacional y será posible usarlo para futuras

investigaciones enfocadas en el comportamiento de la sociedad Colombiana.

8. REFERENCIAS BIBLIOGRÁFICAS

[1] Miranda R. Andrus, Pharm. D., and Mary T. Roth, Pharm.D. “Health Literacy: A Review”. Pharmacotherapy 2002;22(3):282–302.
[2] Temple University Hospital. “Electroencephalography (EEG) Resources”. [Online]. Available at: https://www.isip.piconepress.com/projects/tuh_eeg/html/downloads.shtml.
[3] San Diego University. “EEG / ERP data available for free public download (updated 2019)”. [Online]. Available at: https://sccn.ucsd.edu/~arno/fam2data/publicly_available_EEG_data.html.
[4] Marshalla. P, Bar-Haimb. Y, Foxa, N. “Development of the EEG from 5 months to 4 years of age”. Clinical Neurophysiology 113 (2002) 1199–1208.
[5] C.S. Herrman and T. Demiralp. “Human EEG gamma oscillations in neuropsychiatric disorders”. Clinical Neurophysiology Volume 116, Issue 12, December 2005, Pages 2719-2733.
[6] Clínica Mayo. “Electrocardiograma”. [Online]. Available at: https://www.mayoclinic.org/es-es/tests-procedures/emg/about/pac-20393913?page=0&citems=10
[7] Clínica Mayo. “Electromiografía”. [Online]. Available at: https://www.mayoclinic.org/es-es/tests-procedures/emg/about/pac-20393913?page=0&citems=10
[8] González, C. “La Informática Médica y los Sistemas de Información”. Santiago, Mayo 2003. [Online]. Available at: http://www.medicinadefamiliares.cl/Trabajos/infosiscgs.pdf.
[9] Kamel, M and Wheeler B. “The emerging Web 2.0 social software: an enabling suite of sociable technologies in health and health care education”. Faculty of Health and Social Work, University of Plymouth 29 December 2006.
[10] Kumar, S., Nilsen, W., Pavel, M., Srivastava, M. “Mobile Health: Revolutionizing Healthcare Through Transdisciplinary Research”. IEEE Computer Magazine 46(1), 25–38 (2013)
[11] Boyland, J. “FOOL 2012 : 19th International Workshop on Foundations of Object-Oriented Languages”. University of Wisconsin-Milwaukee. October 1, 2012
[12] Nkenyereye, L. WookJang, J. “Performance Evaluation of Server-side JavaScript for Healthcare Hub Server in Remote Healthcare Monitoring System”. Procedia Computer Science, Volume 98, 2016, Pages 382-387
[13] Madhuri A. and Anushree, D. “Single Page Application using AngularJS”. Madhuri A. Jadhav et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (3), 2015, 2876-2879

[14] Angular.io. “Introduction to the Angular Docs”. 2020. [Online]. Available at: https://angular.io/docs
[15] Angular Material UI. “Getting Started with Angular Material”. 2020. [Online]. Available at: https://material.angular.io/guide/getting-started
[16] Ray, P.P., Dash, D. & De, D. “Real-time event-driven sensor data analytics at the edge-Internet of Things for smart personal healthcare”. J Supercomput (2019). https://doi.org/10.1007/s11227-019-03089-w
[17] Node.JS. “Documentación de Referencia de la API”. 2020. [Online]. Available at: https://nodejs.org/es/docs/
[18] Amazon, “Explore our products,” 2020. [Online]. Available at: https://aws.amazon.com/products/
[19] Microsoft. “Azure solutions”. 2020. [Online]. Available at: https://azure.microsoft.com/en-us/solutions/
[20] MongoDB Atlas. “Get Started with Atlas”. 2020. [Online]. Available at: https://docs.atlas.mongodb.com/getting-started/
[12] D. Gachet, M. de Buenaga, F. Aparicio and V. Padrón, "Integrating Internet of Things and Cloud Computing for Health Services Provisioning: The Virtual Cloud Carer Project". 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Palermo, 2012, pp. 918-921.