UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELECTRICA

DESARROLLO DE WEB APP SOBRE TOXICOLOGÍA CLÍNICA DE APOYO A PERSONAL DE SALUD

POR

Gerardo Alonso Castillo Rodríguez

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al grado académico de Ingeniero(a) Civil Biomédico(a)

Profesor(es) Guía
Dra. Pamela Guevara A.
Dr. Claudio Müller R.

Profesor Revisor
Dr. Pablo Aqueveque.

Septiembre 2022
Concepción
(Chile)

Agradecimientos

En primer lugar, quiero agradecer a mi familia, en especial a mis papás Solange y Gerardo, a mi segunda madre Ana Teresa, quienes siempre se han destacado por hacerme los gustos, pero enseñándome que las cosas hay que ganárselas con esfuerzo, por toda la presión que ejercieron en mí y crear involuntariamente una motivación en darles la felicidad de ver a su hijo cumplir sus metas. Gracias por darme los valores que hoy me hacen la persona que soy, esa persona que siempre está queriendo mejorar y llegar más lejos.

A mi cable a tierra, la persona que llegó a mi vida bajo el mismo contexto universitario con el cual realicé este escrito. Eres y serás siempre la chica que me bajaba los humos, que me decía en que me equivocaba y entregaba el punto de vista que nadie quiere dar, pero que siempre se necesita. Me apoyaste como nadie y lograste sacarme de los momentos tristes. Sin dudas esto también es para ti, fuiste esencial en mi desarrollo para llegar a este punto, gracias por bancarme siempre y por estar ahí cuando te necesité.

A mi amigo Francisco e Ignacio, que apañaban en momentos buenos y momentos malos, por todos los consejos que me dieron cuando los necesité. Gracias por ser de esos amigos fieles que duran tantos años y sabes que los puedes buscar para lo que sea.

A mis amigos del discord que son la voz de apoyo en esas noches de estudio prolongado. Las que te sacan un rato del estrés y permiten que se siga avanzando con la misma motivación. Por todas esas risas, todas las conversaciones y los gritos de aliento.

A todos mis amigos de la universidad, que me aclararon dudas cuando las tuve, que me apoyaron y estuvieron de muy cerca durante todo este proceso. Gracias por su amistad y por sobre todo por tener una enorme sonrisa hasta en los peores días de estrés.

Agradezco a los profesores toxicólogos Berta Schulz de la Universidad de Concepción y Luis González de la Universidad de Magallanes, por haber aportado con sus comentarios y retroalimentación acerca de la web app desarrollada.

Agradecer de igual manera a los docentes, Pamela Guevara y Claudio Müller por su disposición, comentarios y por la confianza que pusieron en mí para desarrollar este proyecto.

Por último, a todos aquellos que de alguna u otra forma hoy forman parte de esto, esta victoria también es para ustedes.

Resumen

En este documento se entregan los aspectos fundamentales del desarrollo de una aplicación web para abordar la implementación de una herramienta capaz de ofrecer información valiosa sobre toxicología clínica, ya que, los actuales mecanismos para obtener esta data resultan poco expeditos para el personal de salud y/o público en general.

Se tomaron los datos reunidos en una tesis de Química Farmacéutica de la Universidad de Concepción y se introdujeron a una base de datos. Dentro de los datos reunidos se encuentra el 10% de las intoxicaciones más recurrentes en Chile según números del Departamento de Estadísticas e Información de Salud (DEIS).

Mediante las tecnologías de MERN, utilizando Visual Studio Code se logró crear una web app subida a la plataforma Netlify. Esta app es capaz de buscar y mostrar información directamente desde una base de datos subida en la nube de MongoDB Atlas. En ella se muestran datos sobre Tóxicos y Antídotos en categorías como descripción, posología, tóxico cinética, dosis tóxica, fase de síntomas, descontaminación y modo de uso, según corresponda a un tóxico o antídoto. Se logró la implementación de un recuadro de búsqueda conectado a un servidor montado en Heroku para buscar toda la información. Se creó una subpágina con un formulario de contacto para recibir dudas de los usuarios, como solicitar más información referente a los tóxicos.

Se evalúo la funcionalidad de la web app mediante una encuesta hacia profesionales relacionados al área de la toxicología clínica. Los resultados indicaron que la función principal se cumple con éxito y sería una plataforma utilizada por buena parte de los encuestados. A su vez estos comentaron que se podría implementar una sección de noticias sobre tóxicos. Con esto se concluyó que los objetivos planteados se consideran cumplidos con éxito con posibilidades de agregar nuevas funciones y/o mejorar la parte estética de la web app.

Abstract

This document provides the fundamental aspects of the web application development to aboard the implementation of a tool capable of offering valuable information on clinical toxicology, since the current mechanisms to obtain this data are not expedient for health personnel and / or public.

The data collected in a Pharmaceutical Chemistry thesis at the University of Concepción was taken and entered to a database. Among the data collected is 10% of the most recurrent intoxications in Chile according to data from the Department of Health Statistics and Information (DEIS).

Through MERN technologies, using Visual Studio Code, it was possible to create a web app uploaded to the Netlify platform. This app can search and display information directly from a database uploaded to the MongoDB Atlas cloud. It shows data on Toxics and Antidotes in categories such as description, dosage, kinetic toxics, toxic doses, phase of symptoms, decontamination, and mode of use, as appropriate to a toxic or antidote. The implementation of a search box connected to a server mounted on Heroku to search all the information was achieved. A subpage was created with a contact form to receive questions from users such as requesting more information regarding toxics.

The functionality of the web app was evaluated through a survey of professionals related to the area of clinical toxicology. The results indicated that the main function is done successfully, and it would be a platform used by a large part of the respondents. In turn, they commented that a news section on toxics could be implemented. After this, it was concluded that the proposed objectives are considered successfully fulfilled with the possibility of adding new functions and/or improving the aesthetic part of the web app.

Tabla de Contenidos

Lista de Tablas

Lista de Figuras

Nomenclatura

Minúsculas

kg	: kilogramos.
mg	: miligramos.
ml	: mililitros.
min	: minutos.
ms	: milisegundos.
px	: píxeles.

Mayúsculas

kB	: kiloByte.

Abreviaciones

Minúsculas

json : Objeto en javascript del tipo documento.
app : Aplicación.
db : Base de datos.

Mayúsculas

PC : Computadora personal.
MYSQL : Lenguaje de consulta estructurado.
DEIS : Departamento de Estadísticas e Información de Salud.
SPA : Aplicación de página única.
MVC : Modelo vista controlador, estilo de arquitectura de software.
JS : Lenguaje de programación javascript.
PHP : Procesador de hipertexto.
HTML : Lenguaje de marcado de hipertexto.
MPA : Aplicación de varias páginas.
C# : Lenguaje de programación C sharp.
MVVM : Patrón de arquitectura de software modelo vista vista modelo.
CITUC : Centro de Información Toxicológica de la Universidad Católica.
SSR : Aplicación renderizada por el lado del servidor.
XML : Lenguaje de marcado extensible.
API : Interfaz de programación de aplicaciones.
PaaS : Plataforma como servicio en línea.
DOM : Modelo de objeto de documento o estructura de documento HTML.
IPS : Proveedor de servicio de internet.
CORS : Intercambio de recursos de origen cruzado.

Introducción

1.1. Introducción General

En Chile cada año se reciben aproximadamente 16 millones de urgencias en los centros asistenciales. Considerando las necesidades actuales del personal de salud, de otorgar una rápida y segura atención, y las ventajas que ofrece la tecnología, surge la necesidad de una herramienta web que entregue información sobre el manejo oportuno de las intoxicaciones agudas en los centros de salud. Se parte de la base que en Chile, tan solo en el año 2019, se registraron cerca de 205 mil casos de intoxicaciones en urgencias, según información de la DEIS [1]. Por esto, se propone desarrollar una web app de toxicología clínica que entregue información al personal de salud contemplando los ítems más importantes respecto a cada tipo de intoxicación. De esta manera, se busca dar a conocer el manejo de las intoxicaciones más recurrentes en los centros de salud utilizando una aplicación creada con mecanismos modernos e intuitivos [1], [2].

1.2. Definición del Problema

El principal problema yace cuando el personal se ve enfrentado a situaciones de urgencia y no tiene la información a la mano o al buscarla debe filtrarla demasiado, cuando debería ser de fácil acceso. Actualmente, el personal de salud al enfrentarse a una intoxicación recurre al número telefónico de emergencias del Centro de Información Toxicológica de la Universidad Católica (CITUC), el cual cumple el rol de entregar la información referente a cada caso. Por lo tanto, actualmente existe una dependencia a este centro. Es por esta razón que resulta imperativo suplir este vacío mediante una Aplicación Web de fácil y rápido acceso.

1.3. Objetivos

1.3.1 Objetivo General

Desarrollar una web app intuitiva con una base de datos de información de toxicología clínica para personal de salud, con información sobre el manejo de las intoxicaciones agudas más recurrentes en Chile.

1.3.2 Objetivos Específicos

(i) Diseñar la web app y base de datos, considerando contenidos a desplegar y las funcionalidades por ofrecer.

(ii) Implementar la base de datos y la web app, enfocadas en su fácil uso por parte del personal de salud.

(iii) Evaluar y optimizar el funcionamiento de la web app en todos sus aspectos.

1.4. Alcances y Limitaciones

Este proyecto contempla el diseño e implementación de una aplicación web para la entrega de información pertinente a intoxicación aguda. En primera instancia, se busca ordenar toda la información perteneciente a los casos más frecuentes de intoxicación en Chile para posteriormente agregarla a una base de datos en colecciones de formato json. Esta estará limitada al 10% de todos los tipos de tóxicos y antídotos existentes.

Para el desarrollo de la web app se usará MERN Stack, que es un conjunto de tecnologías que consta de MongoDB, Express JS, React JS y Node JS como sus componentes. La aplicación web permitirá acceder a menús desplegables que incluyen Tóxicos, Antídotos y Toxíndromes. Estará definida para mostrar información de alrededor de 7 características importantes para cada uno.

Por otra parte, se requiere de cualquier tipo de dispositivo con navegador y conexión a internet. Para hacer pública la web app se necesitó un servicio de hosting para API y app.

1.5. Metodología

Se creó un script utilizando excelToJson para convertir la hoja de cálculo con la base de datos en archivos de formato json. Esta información fue cargada con un software gestor de bases de datos no relacionales llamado MongoDB Compass, el que cumplió con los requisitos simples exigidos para una base de datos como la que se implementó.

El software principal para la elaboración de la aplicación web es Visual Studio Code 1.62, mientras que Node JS propició el entorno de ejecución Javascript y las herramientas necesarias para extender los usos de HTML para realizar una web con vistas dinámicas. Además, se usaron dos bibliotecas de funciones para desarrollo web. La primera, llamada React JS, permitió facilitar el desarrollo del Frontend y optimizó los tiempos para crear interfaces de usuario, y la segunda biblioteca de módulos, llamada Express JS, se encargó del Backend y la conexión con la base de datos.

Para el desarrollo, se consideró usar una Single Page Application (SPA), un método muy utilizado para crear aplicaciones que requieren cambiar solo las vistas, sin tener que recargar por completo la nueva página. Al no necesitar cargar nuevas páginas, este tipo de aplicaciones resultan ser más rápidas y no envían tantas solicitudes al servidor como las páginas más tradicionales [3].

Se programó y desarrolló el lado del servidor y el lado del cliente, montado en la red de área local. Se procedió a realizar la exportación del código para posteriormente subirlo a Heroku en el caso de la API (Backend) y a Netlify en el caso de la app (Frontend).

Finalmente, se realizó una encuesta de evaluación de la web app hacia profesionales del área de la salud relacionados específicamente con Toxicología Clínica.

Revisión Bibliográfica

2.1. Introducción

Actualmente el uso de la tecnología ha ido en aumento sobre todo en este período pandémico del último tiempo. Según el Centro de Estudios Digitales de Fundación País Digital, desde el año 2017 al año 2020 hubo un aumento desde el 72,7% al 80% en los usuarios que hacen uso de internet en nuestro país [2].

Hoy en día con una pequeña búsqueda en internet podemos hallar información sobre un tema en específico como la Toxicología Clínica. Sin embargo, esto supone una tarea extra que es filtrar manualmente la información encontrada. Resulta perentorio atender la necesidad de reunir todo en un sitio web de fácil ingreso y que contenga lo que precisa el usuario.

Cuando se desarrollan páginas webs, es esencial conocer su finalidad puesto que esto decidirá cuál de las decenas de entornos de desarrollo es el indicado para el proyecto. Actualmente, muchos programas y servicios se presentan en dos modalidades, como web app o native app. Una web app se carga en el servidor web y se ejecuta en el navegador, no requiere ninguna instalación a diferencia de la native App. El espectro de aplicaciones web es muy grande, yendo desde pequeñas herramientas hasta software de cálculo matemático con simulación de gráficos, pasando por las adaptaciones de conocidos programas, como servicios de mensajería instantánea como WhatsApp web [4].

En Chile, un medio digital como el que se quiere implementar no existe. La mayoría de los centros asistenciales, al momento de enfrentarse a casos de intoxicación recurre a teléfonos de emergencias para pedir información de cómo abordar al paciente y tratarlo de manera efectiva.

Basándose en el hecho anterior, este proyecto busca desarrollar una aplicación web para el personal de salud que destaque por una interfaz sencilla y que incluya información sustancial para ocuparse de estos casos.

2.2. Toxicología

2.2.1 Definición

La toxicología se define tradicionalmente como "la ciencia de los venenos". Con el tiempo, nuestra comprensión de cómo varios agentes pueden causar daño a los seres humanos y a otros organismos ha aumentado. Esto ha dado como resultado una definición más descriptiva de toxicología como "el estudio de los efectos adversos de los agentes químicos, físicos o biológicos en los organismos vivos y el ecosistema, incluida la prevención y mejora de dichos efectos adversos ". Así lo ha definido el programa de tutoría en toxicología y capacitación para el desarrollo de habilidades fundado por el National Institutes of Health de Estados Unidos [5].

Estos efectos adversos pueden tomar muchas formas, desde la muerte inmediata hasta cambios sutiles que no se aprecian hasta meses o años después. Pueden ocurrir en varios niveles dentro del cuerpo, como un órgano, un tipo de célula o un bioquímico específico. Nuestra comprensión de cómo los agentes tóxicos dañan el cuerpo ha progresado junto con el conocimiento médico. Ahora sabemos que varios cambios observables en las funciones anatómicas o corporales en realidad son el resultado de cambios previamente no reconocidos en sustancias bioquímicas específicas del cuerpo.

Es importante diferenciar a qué corresponden los Tóxicos, las Toxinas y los Venenos (ver Tabla 1. Características Tóxicos, Toxinas y Venenos)

Tóxicos	Toxinas	Venenos
-Sustancias que producen efectos biológicos adversos de cualquier tipo. -Puede ser de naturaleza química o física. -Los efectos pueden ser agudos o crónicos.	-Péptidos o proteínas producidos por organismos vivos. -Los venenos son toxinas inyectadas por mordedura o picadura.	-Toxinas producidas por organismos.

Tabla 1. Características Tóxicos, Toxinas y Venenos.

2.2.2 Toxíndromes

Una búsqueda por clasificar al conjunto de signos y síntomas ocasionados por un tóxico acuñó el concepto Toxíndromes o Síndromes Toxicológicos. Se pueden caracterizar por síndromes, fármacos ocasionados, estado mental que conlleva, efectos en las pupilas, frecuencia cardíaca/respiratoria y su cuadro clínico (Ver Figura 1. Síndromes Toxicológicos).

Estos permiten hacer una primera apreciación en cada caso de toxicología clínica. Cabe mencionar que los pacientes pueden presentarse con toxíndromes mixtos o parciales porque en la mayoría de los casos no se da el tiempo necesario para que se desarrolle el cuadro completo.

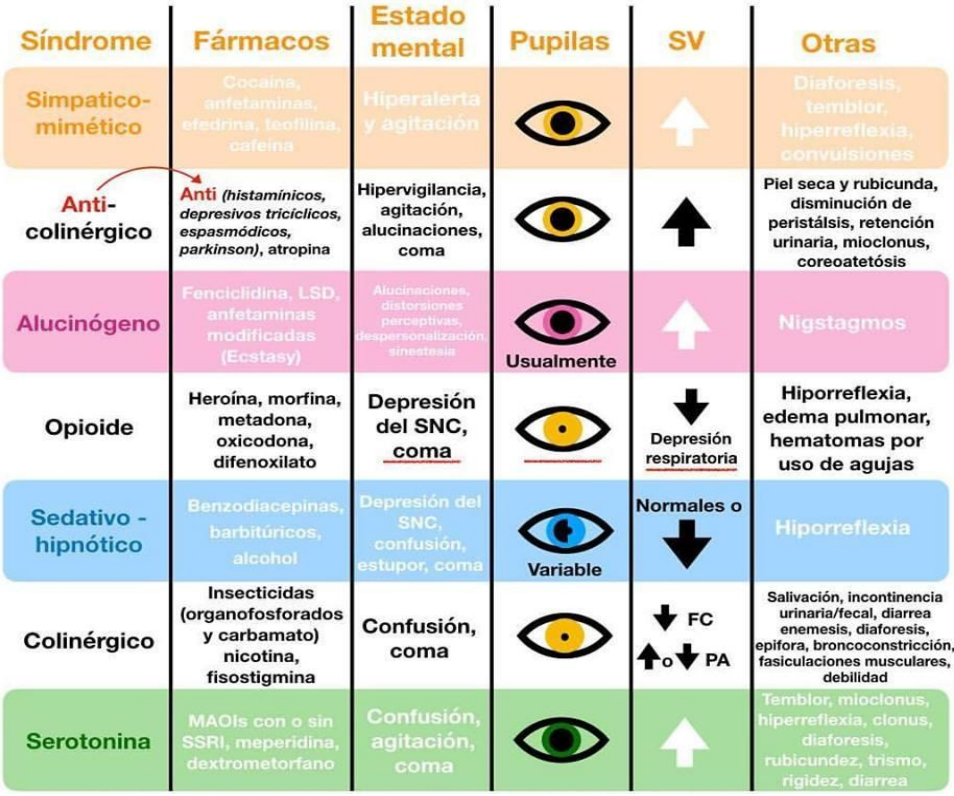

Figura 1. Síndromes Toxicológicos [5].

Se pueden apreciar los efectos que genera cada síndrome, ya sea, estado mental, cambio en las pupilas, signos vitales y otros más efectos.

Mediante el diccionario médico de la Clínica Universidad de Navarra, España, se extrajeron las concepciones más relevantes que involucran los estudios de tóxicos [6]:

- Agente tóxico: Es cualquier sustancia capaz de producir un efecto nocivo en un organismo vivo.
- Posología: Estudia la dosificación de los medicamentos, tanto de la cantidad de medicamento como del intervalo de tiempo entre las administraciones sucesivas.
- Toxicocinética: Estudia los cambios que ocurren a través del tiempo durante la absorción, distribución, biotransformación y eliminación de una sustancia tóxica en el organismo. En breves palabras se relaciona con la dosis.
- Toxicodinámica: Estudia los efectos tóxicos y los mecanismos de acción de los agentes químicos o físicos sobre el organismo. En otras palabras, se relaciona con los efectos.
- Dosis: Es la cantidad de una sustancia administrada en un instante de tiempo.
- Signos clínicos: Son las manifestaciones objetivas, clínicamente fiables, y observadas en la exploración médica, es decir, en el examen físico del paciente.
- Síntomas: Son manifestaciones de elementos subjetivos, señales percibidas únicamente por el paciente como, por ejemplo, el dolor, la debilidad y el mareo.
- Antídoto: Sustancia o medicamento que sirve para neutralizar o contrarrestar los efectos de un veneno o de un agente tóxico.
- Xenobiótico: Es una sustancia química que se encuentra dentro de un organismo que no se produce naturalmente o se espera que no esté presente dentro del organismo.

2.2.3 Efectos tóxicos

Muchos factores juegan un papel potencial en la toxicidad. La dosis (o cantidad de exposición) es el factor más importante. "Todas las cosas son veneno y nada es sin veneno; solo la dosis hace que una cosa no sea un veneno" es un reconocido dicho atribuido al famoso alquimista Paracelso del siglo XVI quien ya hacía hincapié a este principio, en donde cualquier sustancia puede ser un medicamento o un veneno, sólo la dosis determina una u otra cosa [7], [8].

Los productos químicos pueden causar muchos tipos de toxicidad mediante una variedad de mecanismos. Algunos actúan localmente, como cuando la exposición directa desencadena irritación de la piel o los ojos, mientras que otros químicos causan efectos sistémicos en el cuerpo en lugares alejados de donde ocurrió la exposición real. La toxicidad puede actuar afectando directamente a componentes subcelulares, como los receptores celulares, o puede causar problemas a nivel celular, como por ejemplo con exposiciones a sustancias cáusticas o corrosivas.

2.2.4 Dosis

Dosis por definición es la cantidad de una sustancia administrada de una vez (en un instante de tiempo) tal como se mencionó anteriormente. Sin embargo, se necesitan otros parámetros para caracterizar la exposición a xenobióticos. Los más importantes son el número de dosis, la frecuencia y el período de tiempo total del tratamiento [5].

Las sustancias pueden ingresar al cuerpo como:

- Encontrándolos en el medio ambiente (exposición).
- Por ingesta o administración de una determinada cantidad de sustancia.

Los entornos en los que están presentes los xenobióticos incluyen aire exterior, aire interior y agua. Los xenobióticos pueden viajar al cuerpo a través de la piel, los ojos, los pulmones y el tracto digestivo.

La exposición a un xenobiótico puede ocurrir en cualquier ambiente donde una sustancia pueda ingresar:

- Piel por absorción dérmica (aire y agua).
- Tracto respiratorio por inhalación.
- Tracto digestivo por ingestión.

Una dosis se puede considerar:

- Una medida de exposiciones ambientales.
- La cantidad de una sustancia administrada durante un período de tiempo.

Los tipos de dosis incluyen:

- Dosis absorbida: la cantidad de una sustancia que ingresó al cuerpo a través de la piel, los ojos, los pulmones o el tracto digestivo y fue absorbida por órganos o tejidos particulares. La dosis absorbida también se puede llamar dosis interna.
- Dosis administrada: la cantidad que se administra generalmente por vía oral o por inyección (tenga en cuenta que una dosis administrada por vía oral puede no necesariamente ser absorbida).
- Dosis total: la suma de todas las dosis individuales.

No todas las sustancias que ingresan al cuerpo son necesariamente absorbidas por este. Este concepto se aplica a la ingesta de agua. Cuando una persona bebe una gran cantidad de agua de una sola vez, parte de ella se absorbe mientras que el resto del agua se elimina.

Los términos para los tipos de dosis ayudan a explicar la cantidad de una sustancia que ingresó al cuerpo por diferentes medios, pero la cantidad absorbida es lo más importante [5].

2.2.5 Factores importantes

El fraccionamiento de una dosis total generalmente disminuye la probabilidad de que la dosis total cause toxicidad. La razón es que el cuerpo a menudo puede reparar el efecto de cada tóxico si transcurre suficiente tiempo antes de que se reciba la siguiente dosis. En ese caso, una dosis total que sería dañina si se recibiera de una vez no es tóxica cuando se administra durante un período de tiempo [5].

Las unidades utilizadas en toxicología son básicamente las mismas que se utilizan en medicina. El gramo [g] es la unidad estándar. Debido a que comúnmente las exposiciones son en cantidades más pequeñas se utiliza la notación científica miligramo [mg].

La edad y el tamaño corporal de una persona afectan los efectos clínicos y tóxicos de una dosis determinada. La edad y el tamaño corporal suelen estar relacionados, especialmente en los niños. Esta relación es importante porque el tamaño del cuerpo de una persona puede afectar la carga que tiene una sustancia sobre ella.

Una forma de comparar la eficacia de una dosis y su toxicidad es evaluar la cantidad de sustancia administrada con respecto al peso corporal. Una medida de dosis común es mg / kg, que representa mg de sustancia por kg de peso corporal.

Otro aspecto importante es el tiempo durante el cual se administra una dosis. Eso es especialmente importante para las exposiciones que ocurren durante varios días o que son crónicas. Debido a que la unidad de tiempo más común es 1 día, la unidad de dosificación habitual es mg / kg / día.

Las unidades de concentración ambiental se expresan como la cantidad de un xenobiótico en una unidad del medio, que puede ser líquido, sólido o aire. La concentración es la cantidad de una sustancia que se encuentra en una cierta cantidad de otra sustancia, como agua, aire, suelo, alimentos, sangre, cabello, orina o aliento [5].

2.3. Trabajos Previos

Actualmente, el desarrollo de aplicaciones web ha ido en auge debido a que se destaca un aumento de la compatibilidad entre las diferentes plataformas y su rápido acceso lo que produce una mayor captación de usuarios. Esto permite al usuario acceder desde cualquier dispositivo, sin tener que instalar o configurar un software.

Al presente en Latinoamérica, existe una aplicación móvil gratuita de toxicología llamada “Toxicología Hoy”, de origen argentino y creada por el Dr. Dadic Francisco Tomás, Médico especialista en Medicina Interna y Toxicología por la Universidad de Buenos Aires [9]. Esta aplicación brinda información acerca de más de 400 sustancias tóxicas, posee una evaluación de 4.5 estrellas y lleva más de 10 mil descargas en la Google Play Store [10].

La aplicación requiere de un registro previo para utilizarse. Una vez dentro, se encuentra una categoría de tóxicos, grupos, antídotos y de síntomas (ver Figura 2. Pantallazo Inicio de App Toxicología Hoy).

Los tóxicos consideran información como: Nombre, Sinónimo, Características, Usos, Grupo, Toxicocinética, Toxicodinámica, Dosis Tóxica, Síntomas de Intoxicación, Diagnóstico, Tratamiento y Nomograma (ver ejemplo de Paracetamol en Figura 3. Ficha Paracetamol).

Figura 2. Pantallazo Inicio de App Toxicología Hoy.

Se aprecia el buscador principal donde se muestran algunas sugerencias y tiene una sección de búsqueda avanzada. La información está dividida en tóxicos, grupos, antídoto y signos y síntomas.

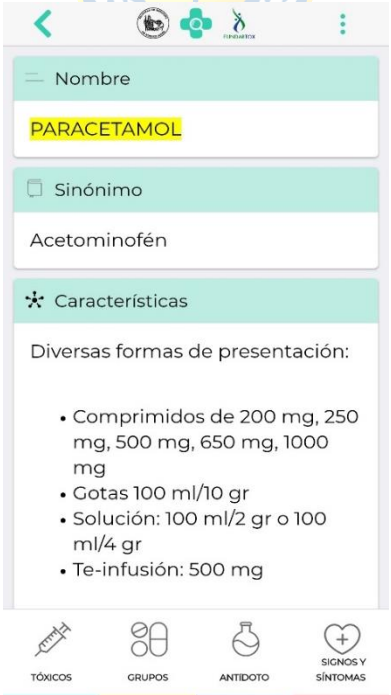

Figura 3. Ficha Paracetamol en App Toxicología Hoy.

En la figura anterior se visualizó como ejemplo el paracetamol que se puede encontrar en diversas formas de presentación y concentración.

Se puede tomar como un excelente ejemplo del proyecto a realizar, ya que, examina puntos relevantes de cada tóxico, lo que nos permite tener una base para la estructuración de los menús y sus respectivos títulos importantes.

Su competencia directa es la aplicación iTox Urgencias Intoxicación, primera aplicación en lengua española que posee información sobre el manejo clínico de emergencias toxicológicas. Contiene más de 2.500 sustancias tóxicas y se adquiere en cualquiera de las tiendas de aplicaciones móviles por un valor aproximado de 16.573 pesos chilenos [11].

En el área de las páginas webs, se puede encontrar ToxicologíaNet, que reúne algunos de los tóxicos más recurrentes [12]. La información se presenta de una manera poco amigable y está dispuesta en un formato de biblioteca (separado por categorías de grupos toxicológicos). El buscador de la herramienta es inestable y tiende a caerse. Reúne una breve introducción, epidemiología, mecanismo de acción, cinética, dosis tóxica, manifestaciones clínicas, diagnóstico, gravedad, tratamiento, bibliografía y una pequeña autoevaluación respecto de los ítems mencionados (ver Figura 4. Paracetamol en ToxicologíaNet).

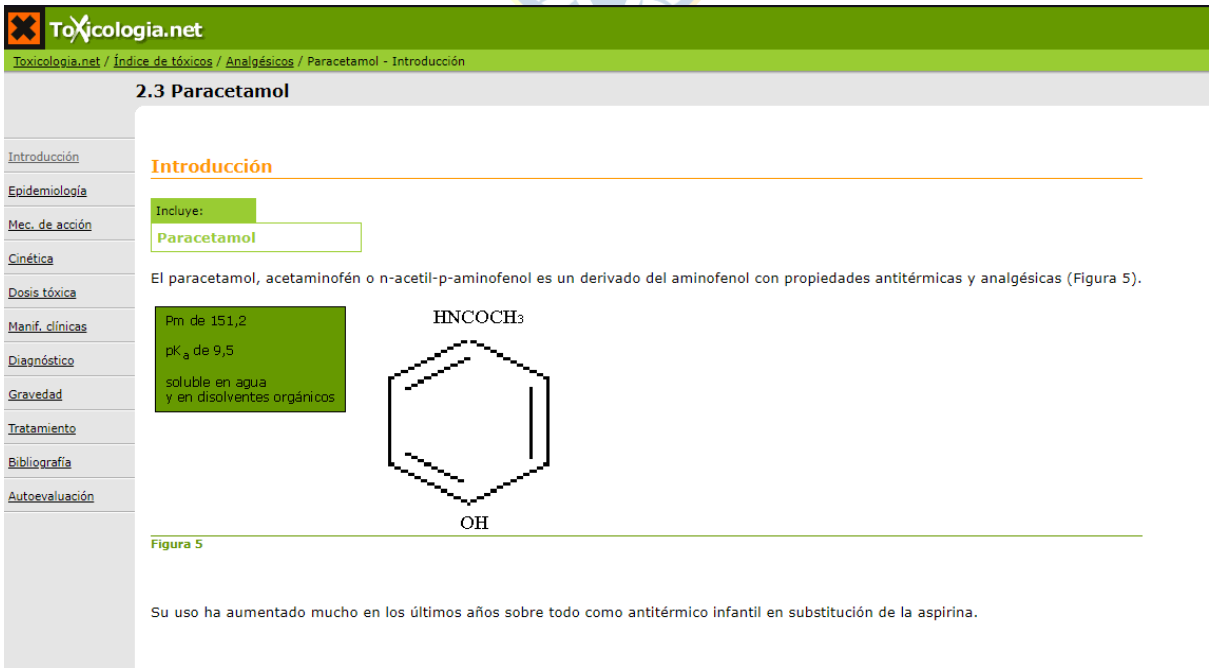

Figura 4. Paracetamol en ToxicologíaNet.

En la figura anterior se visualiza la vista de Paracetamol en la web de Toxicología Net, la cual contiene un índice de tóxicos y variadas secciones de información para cada uno.

Por otro lado, existen blogs con artículos muy variados y que abarcan temas más amplios que solo toxicología, como por ejemplo Patient. Este es un directorio web de información en salud correspondiente a una agrupación de médicos británicos. Contiene más de 4500 artículos distintos donde algunos de ellos corresponden a intoxicaciones comunes y cómo manejarlas [12]

2.4. Tipos de aplicaciones web

Como ya se mencionó con anterioridad, el objetivo principal es que la web app a desarrollar sea fácil, rápida e intuitiva. Para esto se investigaron los distintos tipos de aplicaciones web donde destacan dos tipos [4]:

- SPA: Única página, varias vistas. Funciona cargando el contenido HTML, CSS y JavaScript (Frontend) por completo al abrir la página web. Al ir pasando de una sección a otra, solo necesita cargar el contenido nuevo de forma dinámica si este lo requiere, pero no hace falta cargar la página por completo [13]–[15].
- MPA: Varias páginas, varias vistas. Funciona de manera tradicional, requiriendo que la aplicación se recargue por completo cada vez que un usuario interactúa con ella. Las MPA generalmente tienen datos grandes y arquitectura compleja [13], [14], [16].

Las aplicaciones webs y las páginas en general poseen ciclos de ejecución o llamados lifecycle (ver Figura 5). El ciclo es básicamente la forma en que opera el intercambio de datos entre el cliente y el servidor. Las SPA cambian el ciclo normal porque no utiliza el tradicional metalenguaje XML, sino que utiliza AJAX (Javascript asíncrono y XML) y de esta manera hace uso de objetos de javascript para hacer el intercambio de datos (json) [17]. La principal característica de estos objetos es que tienen una estructura ligera y compacta.

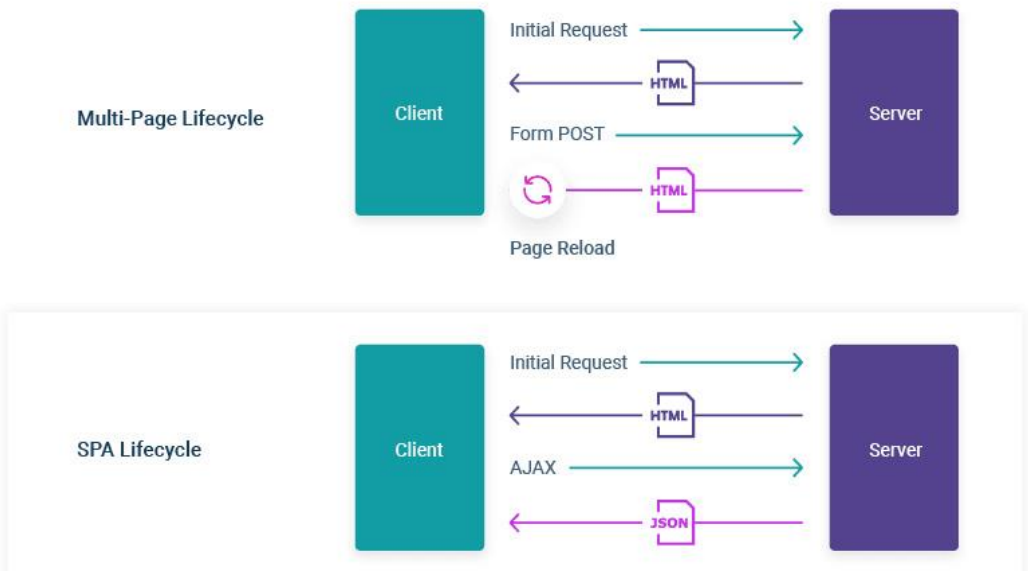

Figura 5. Ciclos de trabajo: MPA vs SPA [13].

Se puede apreciar de la Figura 5 que la principal diferencia entre estos dos ciclos de trabajo es que la MPA se recarga constantemente al generarse solicitudes, mientras que la SPA se carga de una sola vez. Las solicitudes en el primero se realizan mediante Form POST y en el segundo con AJAX.

La arquitectura se puede separar en Frontend y en Backend, que refieren al cliente y al servidor, respectivamente. En otras palabras, el Frontend es la parte a la que un usuario puede acceder directamente mediante una interfaz, mientras que el Backend es la capa de acceso a datos. Esta contiene la lógica de la aplicación y todo lo que se pueda requerir por parte del navegador (Ver Figura 6. Ejemplo de Frontend y Backend) [14], [18].

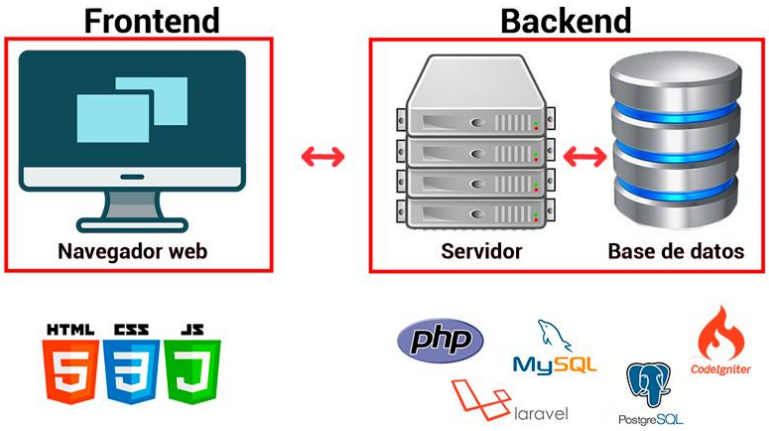

Figura 6. Ejemplo de Frontend y Backend [19].

Para las SPA el Frontend es independiente del Backend, puesto que solo necesitan de una API que les proporcione el contenido [18].

2.5. Bases de datos

Debido a que la aplicación requiere integrar información, lo más preciso es utilizar una base de datos. Es sustancial diferenciar los tipos de bases de datos según su uso. Existen las SQL y NoSQL. La primera refiere a datos relacionales y la segunda a datos no relacionales o documentos. Se considera un documento en una base de datos a un conjunto de campos de información que se asemeja a una ficha y se representa como un objeto Javascript [20].

Para poder ordenar e integrar las bases de datos es necesario un software gestor que facilitará la labor de creación de tablas, crear llaves, establecer relaciones y documentos. El gestor más utilizado por la comunidad para bases de datos relacionales es MySQL mientras que para bases de datos no relacionales es MongoDB y AzureDB [21] (Ver Figura 7. Gestores DB: relacionales y no relacionales).

Figura 7. Gestores DB: relacionales y no relacionales [22]

En el contexto de una base de datos a partir de información que es leída en tiempo real por el Backend existen soluciones en la nube como MongoDB Atlas, medio que ofrece acceso a estos datos con solo una llave generada por su plataforma.

2.6. Entornos de trabajo

El desarrollo web considera el uso de frameworks o en español conocidos como “entornos de trabajo”, los cuales son indispensables para crear un sitio, debido a que proveen una estructura previa. Esto agiliza la labor de implementar una web y al mismo tiempo tienen componentes con un código más limpio, cuyo funcionamiento ya ha sido comprobado [23].

Estos entornos son de distintos tipos y se pueden clasificar por su finalidad principal, aunque algunos son mucho más versátiles, por lo que pueden caber en varias clasificaciones. Dentro de los frameworks podemos destacar:

- Para aplicaciones web
- Para aplicaciones nativas
- Para gestión de contenidos y multimedia

En la siguiente tabla podemos ver algunos de los distintos frameworks disponibles para el desarrollo de aplicaciones web (Ver Tabla 2).

Frameworks	Descripción	Uso principal	Patrón de Arquitectura de Presentación	Lenguaje principal
Angular.js	Un framework basado en *JavaScript* altamente recomendado para la creación de SPA	SPA	SPA	Typescript, Javascript
React	Biblioteca liberada por Facebook, permite desarrollar todo tipo de Apps	Multiplatform Apps	MVC o MVVM o SPA	Javascript
ionic	Biblioteca para móviles, usando HTML, Js, Sass y Angular	Multiplatform Apps	No aplica	Javascript
Meteor	Para desarrollo en *JavaScript*, para web y móviles	Multiplatform Apps	MVC o MVVM	Javascript
Ruby on Rails	Framework basado en Ruby orientado a objetos	MPA	MVC	Ruby, Js
CodeIgniter	Poderoso framework PHP liviano y rápido	MPA	MVC	PHP
Django	Framework Python que promueve el desarrollo rápido y el diseño limpio	MPA	MVC	Pyhton
CakePHP	Framework MVC para PHP de desarrollo rápido	MPA	MVC	PHP
Next Js	Permite funcionalidades de aplicaciones web basadas en React, como la representación del lado del servidor y la generación de sitios web estáticos.	MPA	SSR	Javascript
Catalyst	Framework para aplicaciones web MVC elegante	MPA	MVC	Perl
ASP.NET Core	Es un framework muy versátil que se ejecuta en multiplataforma .NET Core	MPA	MVC, MVVM, Razor Pages	C#, PHP, Js
Laravel	Entorno limpio y con clase para desarrollar PHP	SPA o MPA	MVC	PHP
Vue.js	Perfecto para construcción de interfaces de usuario de una sola página	SPA o MPA	MVC o SPA	Javascript

Tabla 2. Algunos Frameworks o bibliotecas y sus características [24]**.**

La comunidad de desarrolladores constantemente ha estado en busca de facilitar sus labores sin dejar de utilizar sus lenguajes favoritos y esto se puede apreciar en las dos páginas webs más famosas, que son Github (forjador de proyectos) y Stack overflow (sitio de preguntas y respuestas para programadores). Dentro de estos mismos sitios se han hecho encuestas cada año para establecer las preferencias de cada uno basados en sus experiencias, donde se ha podido crear un ranking de los mejores frameworks según su popularidad, ya que, en muchos casos resulta ser una elección netamente de gustos a la hora de seleccionar un framework (Ver Figura 8. Gráfico de HotFrameworks 2016-2021) [24], [25].

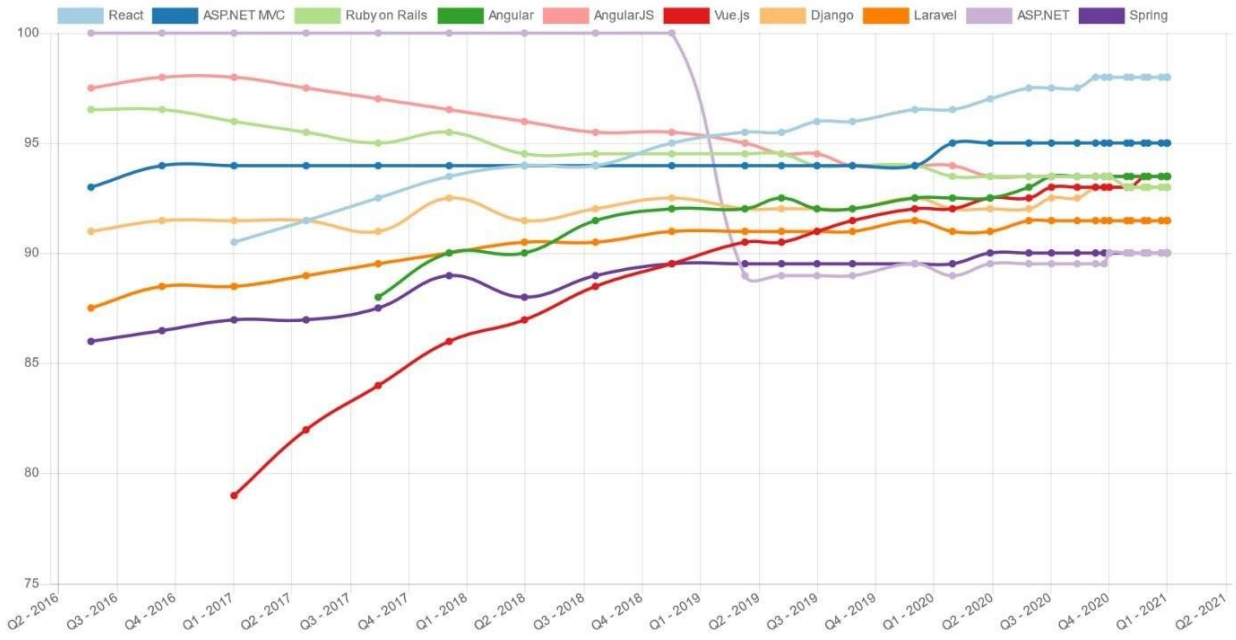

Figura 8. Gráfico de HotFrameworks 2016-2021 [25]**.**

Se necesita una aplicación web con buena apariencia gráfica y para ello existen bibliotecas de CSS, lenguaje principal encargado del diseño gráfico. Los tres frameworks más conocidos son Boostrap, Foundation y Bulma.css.

Actualmente Bootstrap posee muchos tutoriales libres en la red, por lo que lo convierte en un potencial candidato a encargarse de la apariencia de la Web App.

En resumen, las decisiones de diseño más importantes son:

- Definir si será SPA o MPA
- Escoger un framework o Stack para el Backend y para el Frontend
- Escoger base de datos SQL o NoSQL
- Escoger gestor y nube de base de datos
- Biblioteca visual para el Frontend (opcional)

2.7. Herramientas de desarrollo web MERN Stack

En la actualidad uno de los conjuntos de herramientas (Stack) más utilizados en el desarrollo de páginas web es MERN. Tiene todo lo esencial, es extremadamente versátil y cuenta con una amplia documentación y ejemplos.

Básicamente, es un conjunto de los marcos/tecnologías utilizadas para el desarrollo web de aplicaciones, que consta como lo indican sus siglas de MongoDB, Express JS, React JS y Node JS como sus componentes que al unísono logran suplir todas las necesidades para levantar una aplicación web [26].

Definiendo los principales componentes:

- MongoDB: es una de las bases de datos NoSQL más fáciles de utilizar que se conoce para manipular información de texto debido a que está orientada a documentos. Una base de datos MongoDB se puede utilizar para almacenar los datos de la aplicación, cada registro es un documento que consta de pares clave-valor que son similares a los objetos json. Posee MongoDB Compass, un software para revisar visualmente el contenido de las bases de datos. Complementariamente permite utilizar gratuitamente MongoDB Atlas, una herramienta de base de datos en la nube con hospedaje casi permanente.
- Express JS: es un marco para crear el Backend del sitio web con la ayuda de las estructuras y funciones de Node JS. Node JS está destinado a ejecutar JavaScript del lado servidor, pero no para desarrollar sitios web. Express JS está destinado justamente a satisfacer la creación de sitios web.

- React JS: es básicamente una biblioteca que se está utilizando ampliamente para crear componentes de interfaz de usuario, por lo que es relativamente sencillo encontrar distintos ejemplos. Esto puede ayudarnos a crear interfaces de usuario atractivas para nuestras SPA.
- Node JS: Este es un entorno de ejecución para JavaScript que puede permitir ejecutar JavaScript del lado servidor y no en un navegador. Añade la posibilidad de introducir módulos, recursos que pueden ser más o menos simples o complejos en funcionalidad y que contiene un código JavaScript que podemos reutilizar en toda nuestra aplicación. Estos módulos tienen su propio contexto y no interfieren entre sí.

A continuación, se puede apreciar un esquema del funcionamiento de MERN (ver Figura 9).

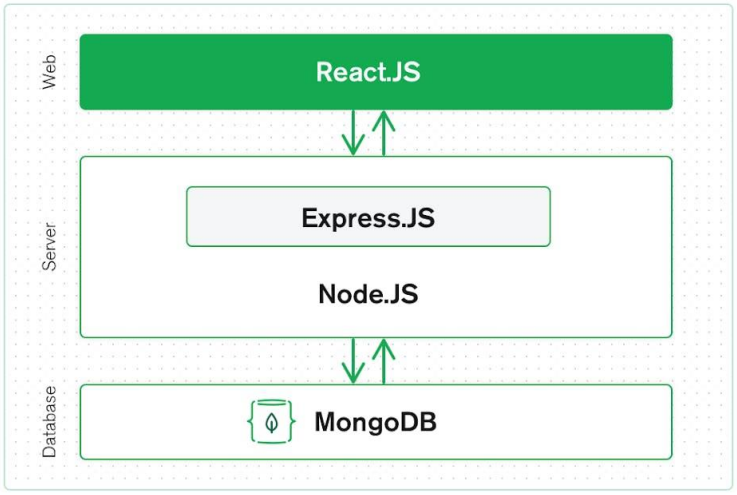

Figura 9. Capas de funcionamiento MERN [27]**.**

Básicamente el Backend está desarrollado en un servidor externo al cliente y la base de datos, pero que sirve como intermediario entre estos dos. React JS se encarga de hacer la representación hacia el cliente web. Express JS y Node JS gestionan tanto las solicitudes desde la base de datos MongoDB como las hacia la misma, que se encuentra alojada en otro servidor.

2.8. Discusión

Como se requiere una página que sea rápida, o sea que no se deba cargar cada vez que se realiza una consulta, y basándose en la literatura, SPA es una excelente opción. Sobre todo, porque requerimos fácil lectura de información ya recopilada. Esto va a aumentar el dinamismo al momento de recorrer los distintos menús, debido a que se considera una experiencia de usuario lineal con las múltiples vistas dentro de la misma página [15], [16].

La principal desventaja de SPA es que, al ser información rotativa, los motores de búsqueda no sabrán ofrecerla a través de los resultados de una búsqueda. A pesar de esta condición, se planea que el nombre de la Web App sea fácil de recordar, y así suplir la desaparición de resultados en materia del contenido ofrecido en la plataforma.

Se puede apreciar un crecimiento de puntuación en los últimos años en React (ver Figura 8) y ha sido bien calificado por los usuarios, llegando a 98 puntos en promedio de 100 en total [25]. Consecuente a este hecho, la cantidad de material disponible para el desarrollo es abundante por lo que es una buena opción de biblioteca para la SPA. Además, opcionalmente se considera el uso de Node JS como motor para Javascript del lado del servidor.

Dado que toda la información a visualizarse tiene forma de tablas u objetos, parece adecuado el uso de una base de datos NoSQL, para así representar estos antecedentes en forma de documentos con el gestor MongoDB [20].

Habiendo mencionado lo anterior, MERN Stack satisface todas las necesidades del proyecto en cuanto al lado del servidor y del cliente.

Todo este desarrollo se llevará a cabo mediante el editor Visual Studio Code con los respectivos complementos para Javascript.

Diseño de la Web App

3.1. Introducción

En este capítulo se detallará el proceso de diseño, estructuración de la Web App, creación de la base de datos e instalación de las bibliotecas necesarias posterior a la amplia revisión bibliográfica y los requerimientos ya establecidos.

Se usó Visual Basic Code como editor de código fuente y consola de depuración, MongoDB Compass como gestor y MongoDB Atlas como nube de la base de datos.

Toda la información toxicológica recogida se exportó en archivos json, un tipo de formato que contiene documentos con una estructura ordenada para que posteriormente sean importados en la nube como colecciones de datos.

Se dividió la Web App en tres secciones, donde se establecieron las bibliotecas involucradas en cada parte y su relación con las demás subpartes. Con esta información se creó una maqueta de la página de inicio de la aplicación web utilizando el servicio de Moqups.

3.2. Requerimientos de información toxicológica

Este trabajo se basó principalmente en un estudio realizado sobre la "Elaboración de las bases para el desarrollo de una aplicación para dispositivos móviles de ayuda en toxicología clínica", realizado por Carla González Norambuena, en su trabajo de fin de carrera presentado en la Facultad de Farmacia de la Universidad de Concepción, para optar al título profesional de Químico Farmacéutico. En este se establecieron ciertos estándares o requerimientos para tener en cuenta a la hora de realizar una aplicación y que serán descritos más abajo [1].

Se enumeran los requisitos de información toxicológica que debe incluir la aplicación:

1) Incluir 5 menús principales: Tóxicos, Antídotos, Toxíndromes, menú de contacto con números de emergencia y de manera opcional un menú de Sobre Nosotros.
2) Se deben añadir los principales síndromes tóxicos o toxíndromes [1]:

- Anticolinérgico
- Colinérgico

- Hipnótico-sedante
- Opioide
- Serotoninérgico
- Simpaticomimético.

3) La aplicación podrá incluir solo el 10% de las intoxicaciones más recurrentes [1] (ver Tabla 3).

Tóxicos
Analgésicos, antiinflamatorios y antipiréticos: -AINEs -Paracetamol
Antialérgicos
Antibióticos: -Penicilinas
Anticoagulantes: -Anticoagulantes orales -Heparinas
Anticonvulsivantes: -Ácido valproico -Carbamazepina
Antidepresivos: -Inhibidores selectivos de la recaptación de serotonina -Tricíclicos
Antineoplásicos
Antipsicóticos
Cocaína
Hipnóticos-sedantes: -Benzodiazepinas
Hipoglicemiantes
Cáusticos
Metales: -Arsénico -Hierro -Mercurio -Plomo
Monóxido de carbono
Plaguicidas organofosforados y carbamatos
Veneno de araña: -*Latrodectus thoracicus* -*Loxosceles laeta*

Tabla 3. Tóxicos a incluir en la Aplicación.

4) La ficha de cada tóxico debe contener información crítica seleccionada de acuerdo con el estudio de otras aplicaciones y la “Encuesta a profesionales de la salud con experiencia en la atención de pacientes intoxicados’’. Esta fue realizada por Carla González a profesionales, con el fin de conocer las preferencias con respecto a la presentación de la información, así como los requerimientos de información inmediata necesaria a la hora de atender estas urgencias (Ver Tabla 4) [1].

Información	
Nombre	
Descripción	
Posología	Niños
	Adultos
Toxicocinética	
Toxicodinámica	Mecanismo de Acción
	Mecanismo de Toxicidad
Dosis Tóxica	Niños
	Adultos
Signos y Síntomas	Fase 1
	Fase 2
	Fase 3
	Fase 4
Tratamiento	Generalidades
	Descontaminación
	Uso de antídoto

Tabla 4. Información a incluir en Tóxicos [1]**.**

Una vez establecidos los menús necesarios y los campos de información requeridos, se dispone de la base de datos en Excel facilitada por la tesis de Carla González, la cual será implementada posteriormente en archivos json.

3.3. Estructuración del proyecto

Se crea un esqueleto del proyecto con todas las funcionalidades que aquí se involucran y los distintos niveles que estas tecnologías tienen. Este se divide en tres partes más importantes, como el Frontend (lado del cliente), Backend (lado del servidor) y, de manera más externa, la Base de Datos (ver Figura 10) [28].

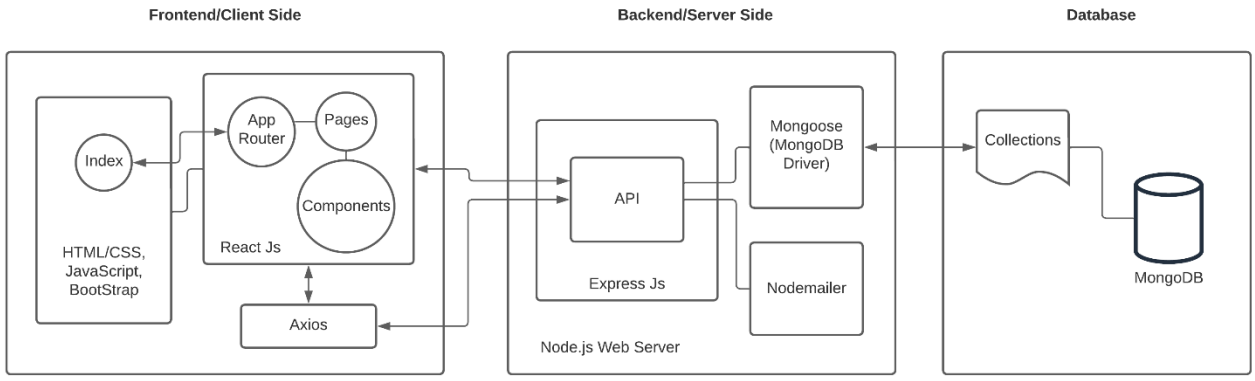

Figura 10. Diagrama MERN SPA con base de datos, envío de formularios y nodemailer.

En la Figura 10 se muestran las divisiones que presenta este tipo arquitectura de aplicaciones web, en donde además se utiliza una base de datos y un intercambio de datos entre cliente/servidor [28]. Esto resume el flujo de información y representa los elementos a utilizar en la aplicación web. A continuación, se describen estas partes.

- Base de datos: En este módulo se crean las colecciones de MongoDB con todos los documentos json pertenecientes a tóxicos, antídotos y toxíndromes.
- Lado del servidor: Aquí se conecta el lado del cliente con nuestra base de datos y cualquier servicio externo. Node JS ejecuta funciones asociadas a las siguientes bibliotecas:
    - Mongoose: Funciona como el controlador de MongoDB, permitiendo generar una conexión segura para leer o escribir en la base de datos [29].
    - Nodemailer: Está encargado de utilizar servicios de mensajería externos. Permite utilizar Gmail, Outlook y otros servicios [30].
    - Express JS: Permite crear el API capaz de conectar todo el Backend y la base de datos con el lado del cliente [31].
    - Heroku: Permite construir y subir la versión exportada de la API hacia la plataforma de Heroku [32].

- Lado del cliente: Dispone mayoritariamente de las bibliotecas encargadas de recibir interacción del usuario con su navegador y de la representación visual para el mismo.
    - Axios: Se encarga de enviar solicitudes hacia la API (enviar formularios o hacer peticiones) [33].
    - React JS: Genera toda la interfaz gráfica para el cliente y posee subpartes como un enrutador, las componentes y las páginas que se muestran. Las componentes son las distintas secciones que en su unión arman una página (cabecera, cuerpo, pie de página, etc) e involucran las funciones y divisiones de cada apartado de la Web App como un único elemento. Las páginas se encargan de tomar las componentes para renderizarlas según el orden y estilo dispuesto. Por otro lado, para que el resultado final de todo el desarrollo genere una SPA se utiliza un enrutador llamado App, el cual conecta dos o más subpáginas (Inicio, Sobre Nosotros, Contacto), las cuales estarán precargadas durante toda la navegación del usuario [34].
    - Index: El índice o index es el resultado de la interpretación de todas las partes visuales en HTML y CSS. Cabe mencionar que al ser una SPA solo se utiliza un archivo HTML que contiene las demás páginas como JS.

La estructura de la conexión de los elementos de programación del Backend se dividirá en tres principales archivos Javascript como se aprecia en la Figura 11. El archivo “server.js” contendrá la aplicación del lado del servidor [31].

Figura 11. Estructura de la programación del Backend.

Los modelos se conectan al enrutador “Routes” y este se conecta al servidor para estructurar la base de datos por el lado del Backend.

En la Figura 12 se muestra cómo se conectan los distintos elementos que forman el lado del cliente, donde el índice es la sección final. Todas las páginas renderizadas hacen uso de la misma cabecera y pie de página, pero su cuerpo varía según corresponde. La página de inicio utiliza “Hero” y el cuerpo “Base de datos”, la página de contacto usa el cuerpo “Formulario de Contacto”, la página sobre nosotros utiliza el cuerpo “Sobre Nosotros”.

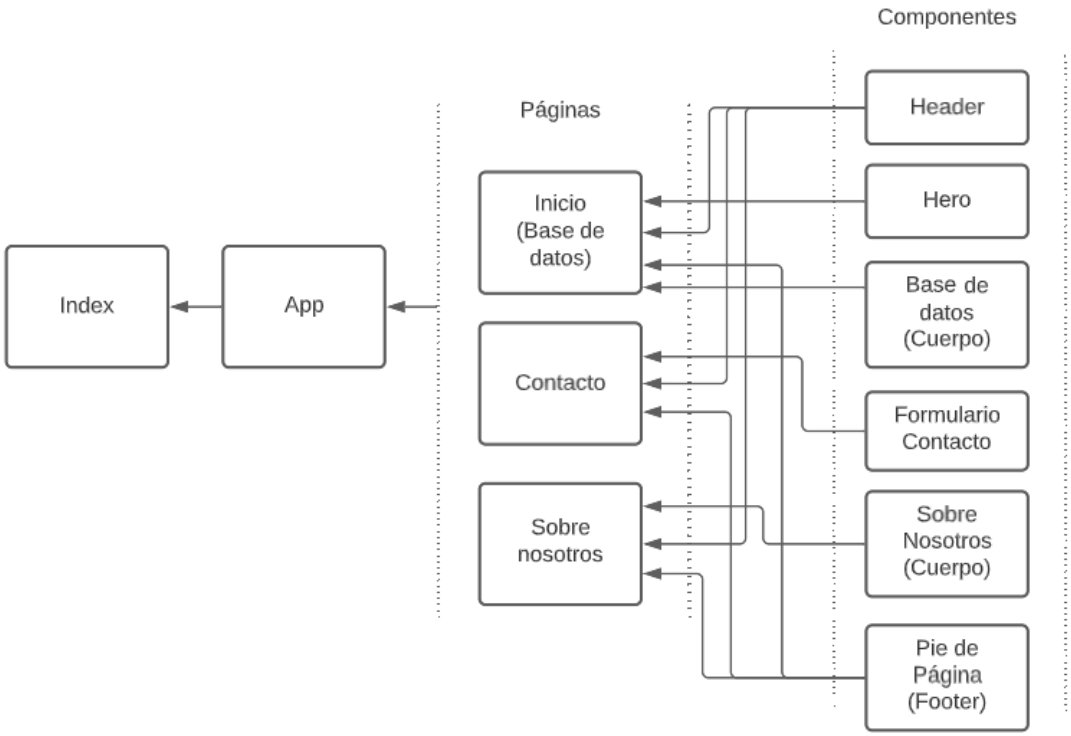

Figura 12. Estructura de la programación del Frontend.

Los principales archivos javascript siguen la estructura anterior, donde las componentes se conectan a las páginas y estas se transforman en la App donde finalmente se renderizan todos estos elementos en el Index.

3.4. Creación de la base de datos

Para importar la información hacia nuestro gestor, primeramente, se debía tener la base de datos en formato json por lo que se abrió el archivo Excel utilizando el conversor “convert-excel-to-json”, una biblioteca de uso simple que permitió exportar los tres archivos requeridos utilizando comandos de javascript [35].

Se crea la base de datos en MongoDB Atlas y se gestiona con Compass mediante la clave asociada a la cuenta adquirida en su plataforma oficial. Este último software permitió cargar la base de datos con todos sus documentos y sus respectivos nombres de colección (ver Figura 13) [36].

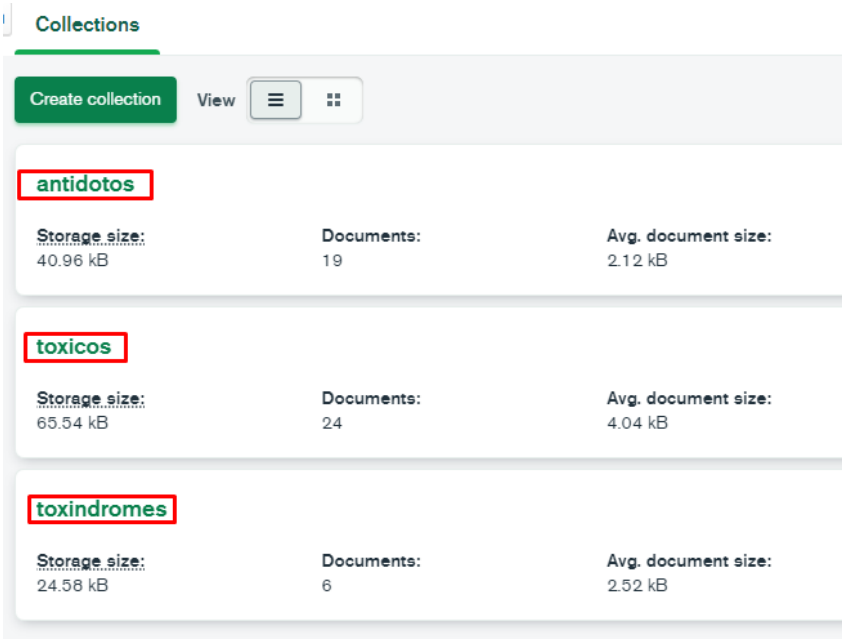

Figura 13. Colecciones MongoDB.

Las colecciones contemplan 19 documentos para antídotos, 24 documentos para tóxicos y 6 documentos para toxíndromes. Al ser solo información de texto utiliza un espacio bastante reducido, ocupando en promedio entre 2 y 3 kB de tamaño por documento y un total de 131 kB.

Es posible ver el nombre asignado a cada fila que posteriormente se utilizó como variables para leer cada uno de los campos (ver Figura 14).

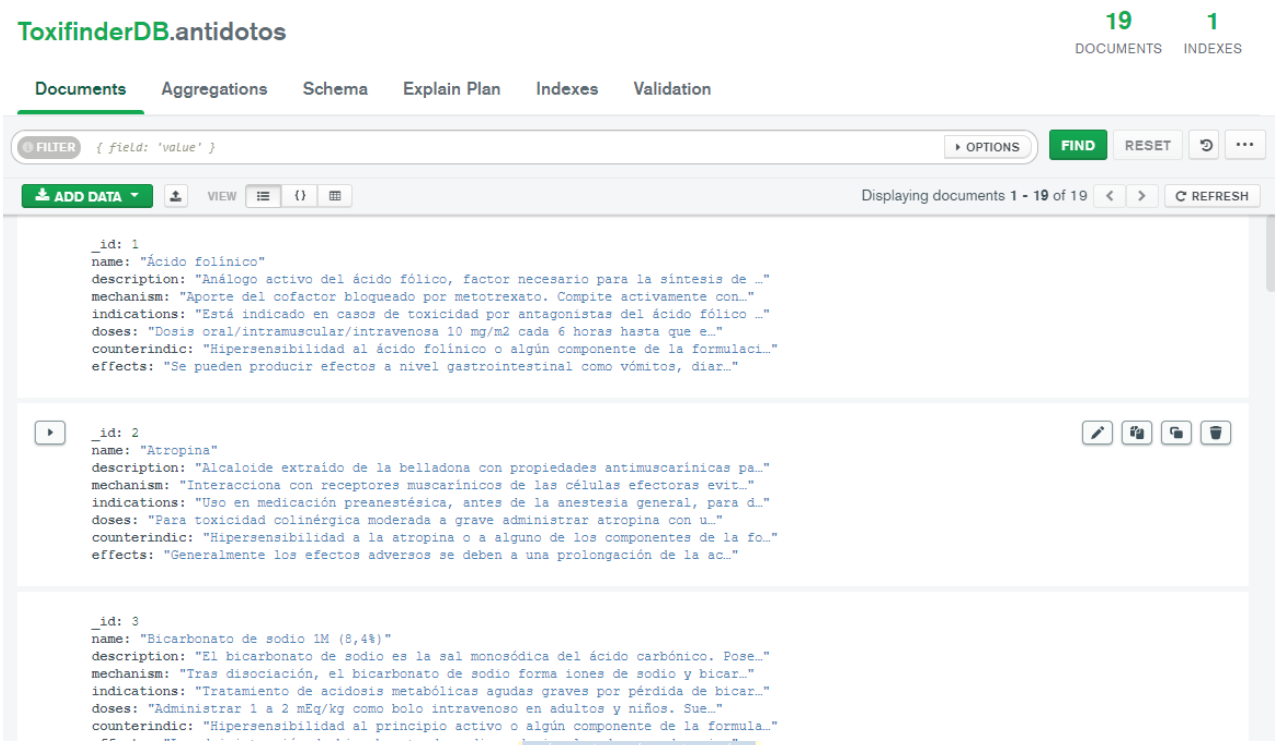

Figura 14. Ejemplo documentos en antídotos.

Los campos que se contemplan son: id, name, description, mechanism, indications, doses, counterindic, effects. Estos se leen como variables por nuestro Backend y se ordenan por su número de id. MongoDB Atlas posee la interfaz mostrada en la figura anterior, para revisar cada uno de los documentos y da la posibilidad de modificarlos directamente.

3.5. Creación de los sitios en línea

Para visualizar el proyecto final desde cualquier dispositivo con acceso a internet es necesario publicar la Web App en un sitio web que permita alojar aplicaciones. Para ello se utilizan las llamadas plataformas como servicio o PaaS. Estas son un modelo de servicio de computación en la nube ampliamente utilizado para estos fines [37]. Las plataformas a utilizar se describen a continuación.

Netlify: Permite a los usuarios configurar sitios web de manera instantánea. Es ampliamente utilizada por desarrolladores para presentar el Frontend, ya que resulta bastante sencillo actualizar los proyectos a nuevas versiones [38].

Heroku: Ofrece soporte para varios lenguajes de programación como Node.js, PHP. Java o Python. Esta PaaS ejecuta aplicaciones en sistemas virtuales que se pueden personalizar según los requisitos de los usuarios y es particularmente recomendada para API's [32].

MongoDB Atlas: Es un servicio de base de datos de múltiples nubes. Atlas simplifica la implementación y la gestión de bases de datos al tiempo que ofrece la versatilidad que necesita para crear aplicaciones. En la sección 3.3 "Estructuración del proyecto", se explica la implementación de la DB en la nube utilizando esta tecnología, por lo que se omite en este inciso [36].

Anteriormente se describió la estructura del proyecto a nivel general. Aquí, en la Figura 15, se despliega un esquema que representa la conexión de red que se usará y las plataformas de servicio involucradas.

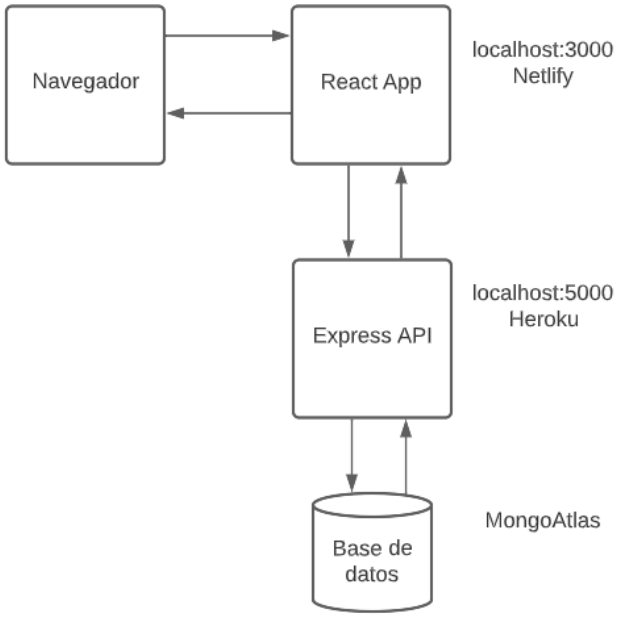

Figura 15. Esquema conexión de red.

Mongo Atlas contempla la base de datos como un elemento externo que se conecta al servidor montado en Heroku utilizando Express (puerto 5000). Este último se conecta al navegador del usuario utilizando la aplicación montada en Netlify por medio de React (puerto 3000).

Para el desarrollo y programación solo se utilizará la red local. No obstante, para su posterior publicación es requerido el uso de estas PaaS, permitiendo de esta forma poder acceder a la web app finalizada desde cualquier dispositivo con navegador y conexión a internet.

Se procedió a crear el sitio bajo el nombre de “Toxifinder” en Netlify y Heroku. Ambas plataformas ofrecen un servicio gratuito de inicio, el que permite crear una aplicación web con un límite de almacenamiento. Netlify otorga el servicio sin límite de tiempo de uso, no así Heroku que luego de cierto tiempo de inactividad se desconecta del servidor hasta que se retome la actividad. Esto se traduce en que si se carga la página cuando Heroku esté en modo inactivo, se tardará más en realizar la conexión nuestro lado del cliente con el servidor.

3.6. Diseño de la interfaz de usuario de la aplicación web

Inicialmente se escogió el nombre de fantasía “Toxifinder” y utilizando el programa de edición Photoshop se creó un logo para la aplicación.

Como ya se estudió, React permite obtener un diseño moderno sin mayores complicaciones. Por esta razón se hará uso de su biblioteca Bootstrap para generar una apariencia legible e intuitiva para el usuario. Considerando las posibilidades que esta entrega se creó una maqueta del diseño estético utilizando Moqups (ver Figura 16), una aplicación que permite generar maquetas web.

Figura 16. Maqueta de la página de inicio con el buscador de la base de datos.

Se aprecia la página de inicio con un Header que se compone de un enlace al Inicio, a la página de Sobre Nosotros, a la página de Contacto y a la página de referencias. Se observa el Hero dividido en dos columnas, donde una tiene información y la otra una imagen. Más abajo se puede ver la parte principal que es el buscador de la base de datos, el que se desea implementar con un acordeón de filas. En la imagen de la Figura 16 se visualizan algunos de los resultados que se esperan si se escriben las letras “An” en el buscador. Finalmente, se ve el Footer compuesto por la marca de Copyright, el logo de la aplicación y sus enlaces a redes sociales.

A partir de esta maqueta se pretende guiar el diseño del Frontend donde se espera utilizar colores sobrios como el gris, blanco y una tonalidad de celeste.

3.7. Discusión

Generar un buen diseño y estructuración de la aplicación web es vital para seguir un orden en la implementación y de esta manera conectar todas las partes, permitiendo encontrar los posibles errores con mayor facilidad y así solucionarlos más rápido.

Se consideró la información necesaria para cada sección de la base de datos para ordenarla en archivos json, la cual fue cargada con el gestor de base de datos Compass. Las colecciones pertenecerán al lado del servidor una vez cargadas desde MongoDB Atlas y los respectivos documentos seguirán el orden ya establecido con sus números de id.

En cuanto a diseño, se presentó una maqueta que utiliza elementos presentes en las bibliotecas de React Bootstrap para facilitar la implementación del lado del cliente.

Cabe mencionar que la estructuración permitió listar todos los recursos a utilizar, como las bibliotecas involucradas en cada componente presentada y cómo estas se conectan con los demás elementos.

Desarrollo de la Web App

4.1. Introducción

En este capítulo se hablará del proceso de implementación y programación en Visual Studio Code de la Web App posterior al diseño y estructuración ya presentada.

Bajo las tecnologías que ofrece MERN Stack, se desarrolló esta aplicación con diversas funciones, donde la principal permite buscar y desplegar información de tóxicos, antídotos y toxíndromes.

Se implementó una sección con un formulario de contacto que genera una respuesta automática por correo, gracias a la conexión del Frontend con el Backend hacia el servicio de mensajería.

Por otro lado, se diseñó toda la parte estética involucrada en el lado del cliente, teniendo como principal objetivo hacerlo intuitivo para el usuario. Toda la información contenida en este capítulo es detallada en mayor profundidad en el Anexo A del presente informe.

4.2. Instalación de bibliotecas

Se instalaron las bibliotecas necesarias para cumplir con la estructura establecida en el capítulo 3.2 “Estructuración del proyecto”.

Para el lado del cliente, se creó un directorio denominado mern-front, donde se instaló boostrap, React JS y axios. Aquí se añadió una carpeta de source o src para los componentes js, css y recursos gráficos.

Por otra parte, para el Backend se creó la carpeta mern-server, donde se dispuso de Node JS, nodemailer, Express JS, Heroku y dotenv. Este último, es un fichero oculto que puede almacenar variables o claves que podemos llamar importándolas en el servidor. Cabe señalar que el Backend es una API que no es del todo segura, por lo tanto, siempre es imperante ocultar la clave de la base de datos mediante dotenv.

4.3. Programación del Backend

Se conservó un orden para la programación del lado del servidor y de esta manera lograr probar en tiempo real el código mediante la consola ofrecida por Visual Studio Code. La ejecución de los comandos se realiza en la carpeta raíz llamada “mern-server”, en la cual se crearon los archivos javascript mencionados en los ítems a continuación.

4.3.1 Modelo para la base de datos

En esta sección se genera el archivo encargado del modelo de la base de datos. Esta tiene como función principal la esquematización de la información a partir de las colecciones de datos. Todo en Mongoose comienza con un Esquema. Cada esquema se asigna a una colección MongoDB y define la forma de los documentos dentro de esa colección [29]. Se definen los campos de información como variables de caracteres (String). Vale decir, se definen como texto variables como: name, description, mechanism, indications, doses, contraindic, effects y otras más correspondientes a los campos utilizados en la base de datos. Se realiza esta operación para los antídotos, tóxicos y toxíndromes (Ver ejemplo en la Figura 1 del Anexo A). Mediante la biblioteca mongoose se anexan los modelos a cada esquema usando la función “mongoose.model” (Ver ejemplo en la Figura del Anexo A) [39]. Luego esto se exporta utilizando “module.exports”, función capaz de extraer objetos, constantes y variables de un archivo JS a otro.

4.3.2 Enrutador del servidor

El enrutamiento o direccionamiento del servidor tiene la función de generar una subpágina para cada una de las tres colecciones de la base de datos, asignando los modelos creados en la sección anterior. Esto se realiza con el fin de cargar los documentos json ya creados. Este proceso se logra gracias a la biblioteca de Express JS que se encarga de todas las solicitudes del Backend [31].

Se creó el archivo “Routes.js” con su función principal homeRouter() la cual contiene todas las rutas de las colecciones, llamadas mediante la función “express.Router()”. En la sección 1.3 del Anexo A se detalla el funcionamiento de homeRouter y cómo se crea el enrutamiento mismo.

4.3.3 Programación de la aplicación del servidor

El archivo más importante del servidor es el “Server.js”, encargado de la ejecución de todo el Backend.

Primeramente, resultó vital definir el puerto de trabajo del servidor debido a que se necesita uno libre para realizar el intercambio de datos por la red [40]. Debido a que se utiliza la PaaS Heroku, el puerto es dinámico y siempre estará definido por la disponibilidad de puertos que la plataforma tenga. Se definió el ruteo utilizando CORS (tipo de control de acceso HTTP) y se realizó la escucha hacia el puerto utilizando la función “express.listen()” [41]–[43] (ver ejemplo de enrutamiento en la Figura 6 del Anexo A).

La conexión hacia la base de datos se realiza con la biblioteca mongoose (ver ejemplo en la Figura 5 del Anexo A). Se emplea la llave facilitada por MongoDB Atlas, la cual es ocultada del código por la biblioteca “dotenv” (carga variables de entorno) [44].

Se programó el servicio para mensajería de nodemailer, donde se analizan las solicitudes entrantes con datos json mediante “express.json()” [45]. Cabe mencionar que se estructuró esta sección para comunicar desde el cliente los datos ingresados en el formulario de contacto y estos se envíen hacia el servidor como documentos json. Estos últimos se enrutan con la función “app.post()” a la ruta especificada como “/users” [33]. En la Figura 17 se aprecia un ejemplo de cómo se envían los datos del formulario de contacto en un documento json.

```
[{"from:toxicologia.farmacia.udec.cl@gmail.com","subject:Prueba1","messsagge:Hola mundo"}]
```

Figura 17. Ejemplo envío de formulario de contacto

Cuando hacemos uso de correos institucionales o poco convencionales normalmente se requiere configurar en nuestra bandeja favorita, como Gmail, iCloud u Outlook. Siguiendo este mismo concepto, se requiere configurar nodemailer con el servicio de mensajería a utilizar. En este caso, se utilizaron las credenciales de un correo de prueba entregado por la plataforma Mailtrap [46], [47]. Además, se definen las variables destinatario, remitente, asunto, contenido y el formato que lleva el cuerpo del mensaje [30], [47]. Para confirmar el estado del envío de este correo, se envía un mensaje hacia el cliente almacenado en la variable “respMesg” siguiendo el mismo concepto de documento json de Express JS (ver ejemplo en el Anexo A, Figura 7).

4.4. Programación del Frontend

Toda la infraestructura visual y funcional del lado del cliente está conservada en la carpeta raíz “mern-front”. Se crearon dos subcarpetas siguiendo la estructura propuesta en el diseño de programación del lado del cliente.

- Pages: Contiene las páginas ya mencionadas en el ruteo, las cuales son About, Contact, Home, Refs.
- Components: Contiene las componentes a utilizar en las páginas, como Header, Footer, Hero, entre otras.

En estas carpetas se generaron los ficheros JS definidos en el ítem 3.2 Figura 12, como sigue a continuación.

4.4.1 Aplicación principal del Frontend

La parte inicial para cargar una web app de arquitectura SPA es su índice, puesto que renderiza la aplicación como tal. Con la biblioteca ReactDOM y su función “render()” se logra crear la página raíz de la aplicación, la cual importa toda la estructura HTML generada por los elementos de la aplicación contenidas en el archivo “App.js” [48]. En este último se realiza el enrutamiento de todo el Frontend mediante la biblioteca React-router-DOM, utilizando la función “BrowserRouter”. De aquí se anexan las demás subpáginas Home, About, Contact y Refs (ver ejemplo en el Anexo A, Figura 8).

4.4.2 Componentes

Tal como se mencionó en la sección de diseño, las componentes son los segmentos que unidos formarán cada página. Comúnmente es fácil identificar la cabecera, el cuerpo y el pie de una página web [49]. Estos elementos mencionados son las componentes de una aplicación web, donde algunos son reutilizados como es el caso del Header y el Footer, segmentos siempre presentes y que tienen la función de facilitar la navegación al usuario a través de la app.

- Header: Este consiste en una barra de navegación anclada a la parte superior de la página y que es obtenida desde la biblioteca React con la función Navbar [34]. Aquí se incluye el logo de la aplicación y los enlaces a las demás páginas. Se estableció que para todos los dispositivos que sean menores a 1200 px, barra de los enlaces se acopla para reducir el espacio en pantalla (ver ejemplo en el Anexo A, Figuras 9 y 10).

- Footer: Es el pie de página, el cual contiene los íconos de redes sociales, el mensaje de Copyright y en el centro un logo horizontal de la aplicación web [50]. Al igual que el Header, se implementa utilizando la función Navbar.
- Hero: Es la sección que primero ve un usuario al entrar en un sitio web. Se dividió en dos partes. En un Hero para poner el mensaje de bienvenida a la web y en otro para visualizar información sobre la base de datos. Se utilizó un Container de React (especie de rejilla para dividir los espacios a utilizar) de una fila y dos columnas para implementar ambas componentes [51].
- DBSearch: Componente más importante para la base de datos que contiene el buscador y la información que se comunica desde el Backend. Aquí se involucraron tres archivos JS para representar los datos de cada categoría a utilizar (Tóxicos, Antídotos y Toxíndromes). React permite hacer lectura de información de una dirección externa con la función "useEffect()". Con esto se realizó el fetch (recuperación de datos) hacia la dirección del Backend otorgada por Heroku. Es decir, se leen las colecciones generadas por el servidor directamente utilizando la función fetch [52] (ver ejemplo en el Anexo A, Figura 13).

  En este apartado se crearon las tres pestañas que separan los Tóxicos, Antídotos y Toxíndromes, usando la función "Tab" de React [53]. Estas pestañas despliegan un acordeón con la lista de documentos correspondiente a su colección [34]. En la parte superior se agregó un buscador que no discrimina mayúsculas ni minúsculas y realiza una constante comparación con los valores almacenados en el nombre de cada documento de la colección (ver ejemplo en el Anexo A. Figuras 15 y 16).
- Refs: Contiene todas las referencias de la base de datos en un Container de única fila con todos los ítems en "ListGroup" (función de React para generar listados de datos en grupos) separados por Tóxicos, Antídotos y Toxíndromes [34].
- About: Correspondiente a la página de Sobre Nosotros. Se utilizó un div para el encabezado y un Container con "Cards" (tarjetas de información) para cada uno de los miembros del equipo del proyecto. Las tarjetas llevan título, subtítulo, foto, texto de presentación y un enlace a más información. En esta componente se integra una mención a los impulsores de la web app, utilizando un Container con logos e información [34].

- Contact: Se utilizó para programar el envío del correo en el formulario de contacto. Mediante la biblioteca Axios se crea una función para enviar la información ingresada por el usuario en el lado del cliente utilizando “axios.post()” (ver ejemplo en el Anexo A, Figura 17) [33], [54]. Esta función recoge los datos almacenados en la constante “user” y los envía a la dirección del Backend.

4.4.3 Páginas

Una vez implementadas las componentes, es necesario unirlas en páginas para dar sentido a la estructura de una aplicación web. A continuación, se describen las páginas generadas y las componentes que involucra cada una.

A. Inicio

Bajo el nombre de “Home.js” se crea la página raíz de la web app. Esta página incluye los componentes de Header, Hero, DBSearch y el Footer.

Paralelamente, en la página de inicio se implementó una ventana emergente de alerta que se muestra al abrir por primera vez durante la sesión la aplicación web. Esta ventana se implementa gracias a Modal de React [34]. Se utilizó la función “sessionStorage()” para establecer el estado de mostrar o no mostrar esta ventanilla dependiendo del estado de la sesión actual del usuario [34], [51], [55]. Esta ventana tiene la finalidad de clarificar al usuario que la información propiciada por la base de datos de la aplicación es de uso exclusivo para personas con debida preparación. Se implementó el Modal con el mensaje: “La información disponible en esta WebApp está dirigida exclusivamente a profesionales y estudiantes de la salud que cuentan con la debida preparación.” (ver ejemplo de su implementación en la Figura 18 del Anexo A).

B. Sobre Nosotros

Esta página contiene toda la información sobre el equipo desarrollador del proyecto (en la componente About). Reúne además las componentes de Header y Footer.

C. Contacto

La página de contacto tiene como finalidad permitir a los usuarios enviar consultas mediante el formulario de contacto (programado en la componente Contact.js) y a su vez también conseguir números de emergencia para ayuda relacionada a toxicología clínica. Esta página además retorna las componentes de Header y Footer.

A. Referencias

Esta página agrega la lista de referencias diseñada en la componente Refs, con la finalidad de otorgar apoyo bibliográfico para la base de datos utilizada en la aplicación.

4.4.4 Archivos multimedia

Para la programación del lado del cliente, las componentes requirieron archivos multimedia. Estos archivos fueron llamados desde una carpeta llamada “images”, donde se incluyó el logo de la Universidad de Concepción, el logo de la Facultad de Química y Farmacia, el logo de la aplicación en una versión horizontal extendida y una versión cuadrada (creado a partir de vectores de uso libre), el logo de la Facultad de Ingeniería, las fotos del equipo detrás de este proyecto y finalmente una caricatura para adornar la página de inicio.

4.4.5 Otros ficheros de la carpeta raíz

Para estilizar a nivel macro se utilizó el archivo “index.css”, donde se definieron los márgenes de la página, la familia de las fuentes, el fondo, ancho, altura y varios parámetros que fueron usados en las diferentes componentes de las páginas [56].

El administrador de paquetes de Node JS genera un archivo con el nombre de “package.json” con todas las versiones de las dependencias instaladas y usadas en el Frontend.

Por otro lado, se creó una carpeta “public” con el archivo “index.html” que enmarca toda la aplicación y contiene los metadatos como el título de la web y el favicon (ícono en miniatura que llevan las páginas al agregar a favoritos, de ahí su denominación) [57].

Finalmente se integra un archivo “_redirects” que tiene la finalidad de solucionar los posibles problemas de direccionamiento web en la plataforma de Netlify [58].

4.5. Build & Deploy

Las aplicaciones web requieren una exportación y una publicación, que reciben el nombre de "build" (construcción) y "deploy" (desplegar) respectivamente. Esto significa que la operación de build es un empaquetado o compilado de código ejecutable listo para usar, mientras que la operación deploy es cuando se configura el código en un entorno junto con sus comandos de arranque en una máquina determinada, o sea, su publicación [59], [60].

Como ya se ha mencionado anteriormente, se utilizaron PaaS distintas para el Frontend y el Backend. El servicio de Heroku realiza el build & deploy de manera automática utilizando una biblioteca extensión de Git denominado Heroku CLI. Este solicita el inicio de sesión en la web antes de realizar cualquier cambio. Mediante el comando push de Git en la consola de Visual Studio Code se realizaron todos los cambios al código [61]. Para el servicio de Netlify se exportó la carpeta del lado del cliente y se importó manualmente a la plataforma [62].

Una vez realizado todo el procedimiento de build & deploy, se consiguió montar la web app en la dirección "https://toxifinder.netlify.app/".

4.6. Discusión

El desarrollo e implementación de la web app se facilitó debido al orden y estructuración previa de la aplicación web. Se instalaron todas las bibliotecas requeridas según el diseño tanto en el Backend como en el Frontend.

Por el lado del servidor fue necesario manipular la base de datos de tal manera que fuera posible su lectura desde el cliente. Para ello se usaron esquemas que son enrutados como archivos json. Además, se implementó la configuración para Nodemailer con un servicio de mensajería de prueba debido a que los servicios de mensajería convencionales no autorizan su uso en aplicaciones externas por políticas de seguridad.

La biblioteca de React-bootstrap otorga funciones que no requieren mayores modificaciones para adaptarse a la interfaz del lado del cliente. Sin embargo, se requirió definir a qué valores la aplicación web se adapta a distintas resoluciones de dispositivo. El buscador de la base de datos en la página de inicio se desarrolló sin distinción de mayúsculas para tener un uso más amigable al usuario.

Finalmente, es importante recalcar el uso de sessionStorage en la ventana de atención, ya que con esta función se logra que la ventana se muestre solo la primera vez que se abre la aplicación web.

Resultados

5.1. Introducción

En esta sección se detallarán los resultados de las distintas vistas y ejemplos de manipulación de la web app posteriormente montada en la red.

Se usó el navegador Microsoft Edge para probar el funcionamiento de la aplicación. Se probaron las distintas resoluciones con el fin de verificar la compatibilidad multidispositivo. Se utilizó el buscador para obtener con algunos tóxicos de ejemplo y así interactuar con los menús disponibles. Se simuló una situación de uso real para revisar los tiempos de respuesta.

Se envió un correo de consulta utilizando la sección de Contacto, con la finalidad de probar la última categoría que tiene disponible la SPA.

Finalmente, se realizó una encuesta de satisfacción a tres profesionales relacionados con el área clínica, haciendo preguntas tanto de apariencia como de su facilidad de uso.

5.2. Vistas y funcionalidad de la web app

5.2.1 Página de Inicio

A. Vista de la ventana emergente

Se visualiza el modal al abrir la aplicación con el mensaje de atención (ver Figura 18).

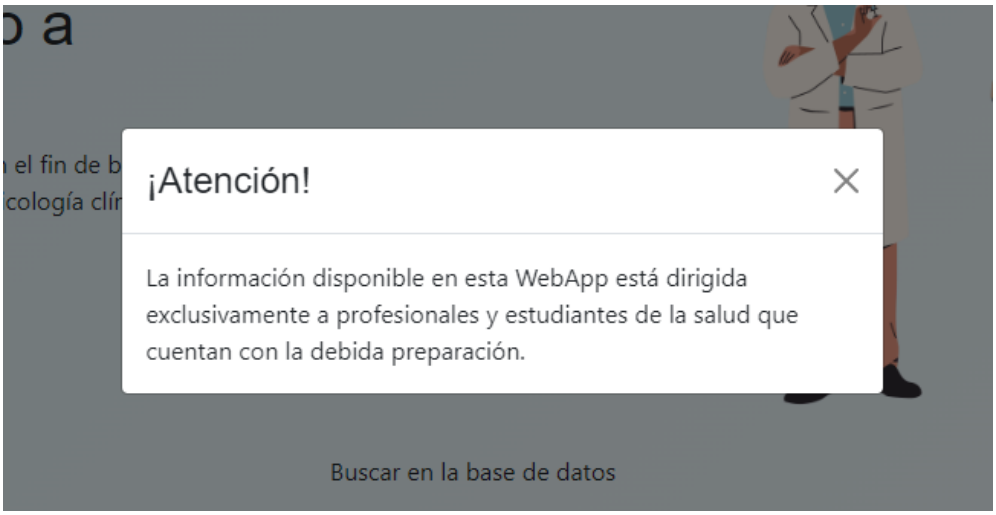

Figura 18. Vista de la ventana emergente de alerta.

B. Vista de la página principal

Se aprecia el header bloqueado en la parte superior de la página, con los respectivos enlaces a las demás secciones de la aplicación web. La parte del Hero se despliega correctamente en dos columnas, tanto en la parte superior de la base de datos como en la posterior. Se puede ver que se renderiza el buscador con el acordeón de Tóxicos, Antídotos y Toxíndromes. El pie de página se muestra con los íconos y la marca de copyright (Figura 19).

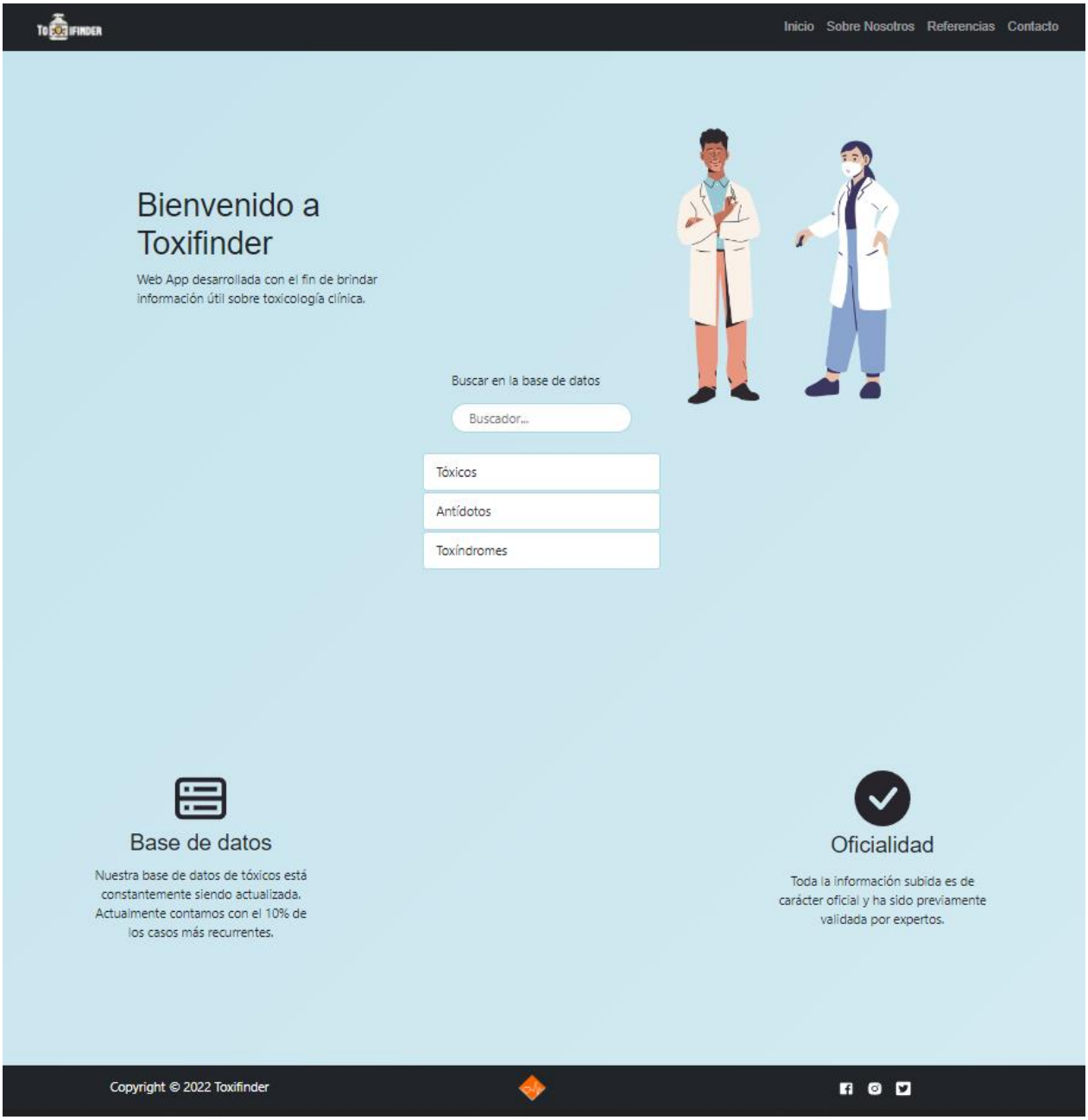

Figura 19. Vista de la página de inicio y todos sus componentes (50% zoom del navegador).

C. Comprobar conexión a base de datos

Para probar la conexión a la base de datos es necesario que se esté ejecutando correctamente el lado del servidor debido a que funciona como un intermediario entre el lado del cliente y MongoDB Atlas. Para ello basta que se muestren los nombres de cada Tóxico, Antídoto y Toxíndromes en el acordeón de la base de datos para así afirmar que la conexión con la base de datos está funcionando con éxito. En la Figura 20 se aprecia que la información fue cargada exitosamente.

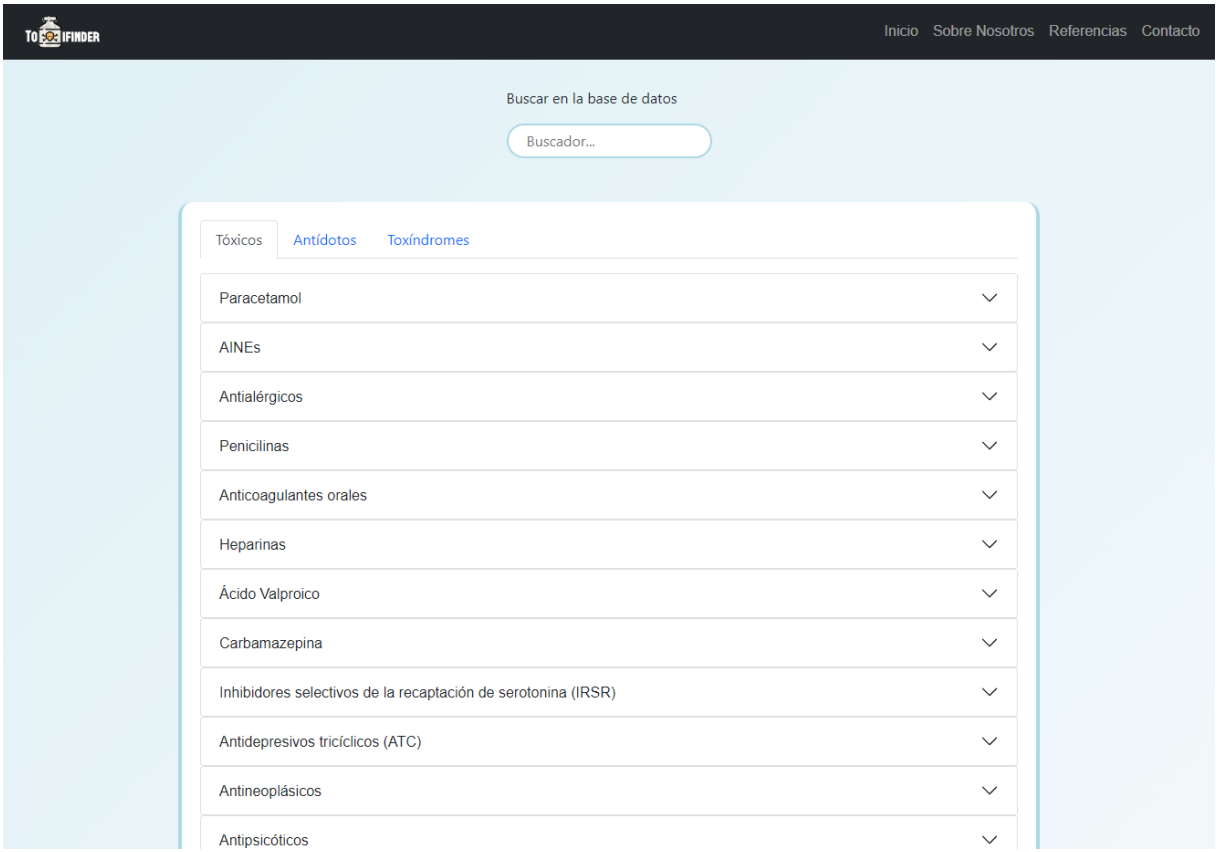

Figura 20. Vista de la base de datos y los menús desplegables

Luego, se demostró el correcto renderizado de las pestañas y los menús desplegables de la base de datos. El botón de “Leer más” despliega correctamente toda la información de cada ítem.

D. Prueba de la ventana emergente

Se realizó una prueba para comprobar el funcionamiento de la ventana emergente de atención denominada “modal”.

En la primera apertura de la página de inicio se desplegó correctamente y se pudo cerrar sin problemas. Al recargar la página no se vuelvió a mostrar, lo cual significa que la funcionalidad de “sessionStorage” se ejecutó correctamente. Al cerrar el navegador para volver a entrar a la web app, se vuelve a mostrar el modal.

E. Evaluación de tiempo de respuesta del buscador

En este inciso se detalla el método y los resultados para evaluar los tiempos de respuesta de la aplicación web. Para realizar esta prueba se utilizaron las herramientas de desarrollador de Microsoft Edge. En la sección de “Red” disponible en el cajón de herramientas se puede apreciar los tiempos de carga de cada elemento y solicitudes generadas [63].

Primeramente, se abrió la herramienta de red para luego entrar directamente en la web app. Se pudo notar que el tiempo de carga total fue de 851 ms (ver Figura 21).

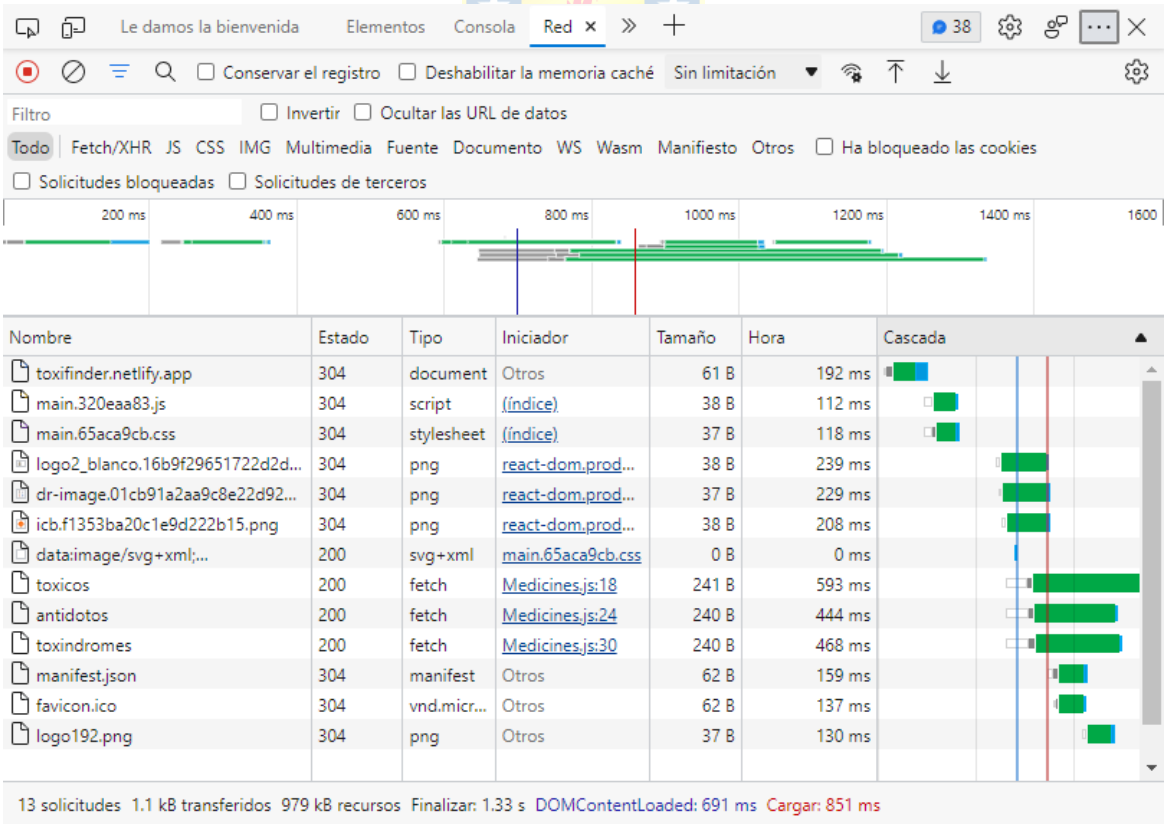

Figura 21. Resultados tiempo de carga elementos de la web app.

Se realizó la búsqueda de Paracetamol en la base de datos para evaluar el tiempo de carga. Se pudo identificar que las variaciones de los tiempo de carga tienen que ver con la calidad de conexión del dispositivo y la velocidad de respuesta de las PaaS utilizadas (Heroku y Netlify). Una vez cargada la página principal, se precarga inmediatamente la base de datos, por lo tanto, la búsqueda de información es instantánea.

Sin embargo, se pudo probar el tiempo promedio de carga utilizando una buena conexión mediante cable de red con Movista Fibra Óptica Hogar como IPS. Se realizaron 5 recargas de la aplicación web usando las teclas “control + R”, dando los siguientes resultados:

N°	Tiempo de carga [ms]
1	851
2	908
3	799
4	707
5	715

Tabla 5. Tiempos de carga de la aplicación web.

Considerando los tiempos de carga listados en la Tabla 5, se obtuvo un tiempo promedio de 796 [ms] para efectuar la carga de la aplicación web.

5.2.2 Página de Contacto

A. Vistas

Se aprecia que la componente principal de esta página se carga correctamente, mostrando por separando en columnas el formulario de contacto y la información de contacto. Si se presiona el botón enviar con los campos vacíos, se envía automáticamente un mensaje de error en gris en la zona del formulario (ver flecha naranja en la Figura 22).

Por otra parte, si se presiona el correo de información de contacto, nos envía directamente a nuestro servicio de mensajería con el destinatario ya establecido.

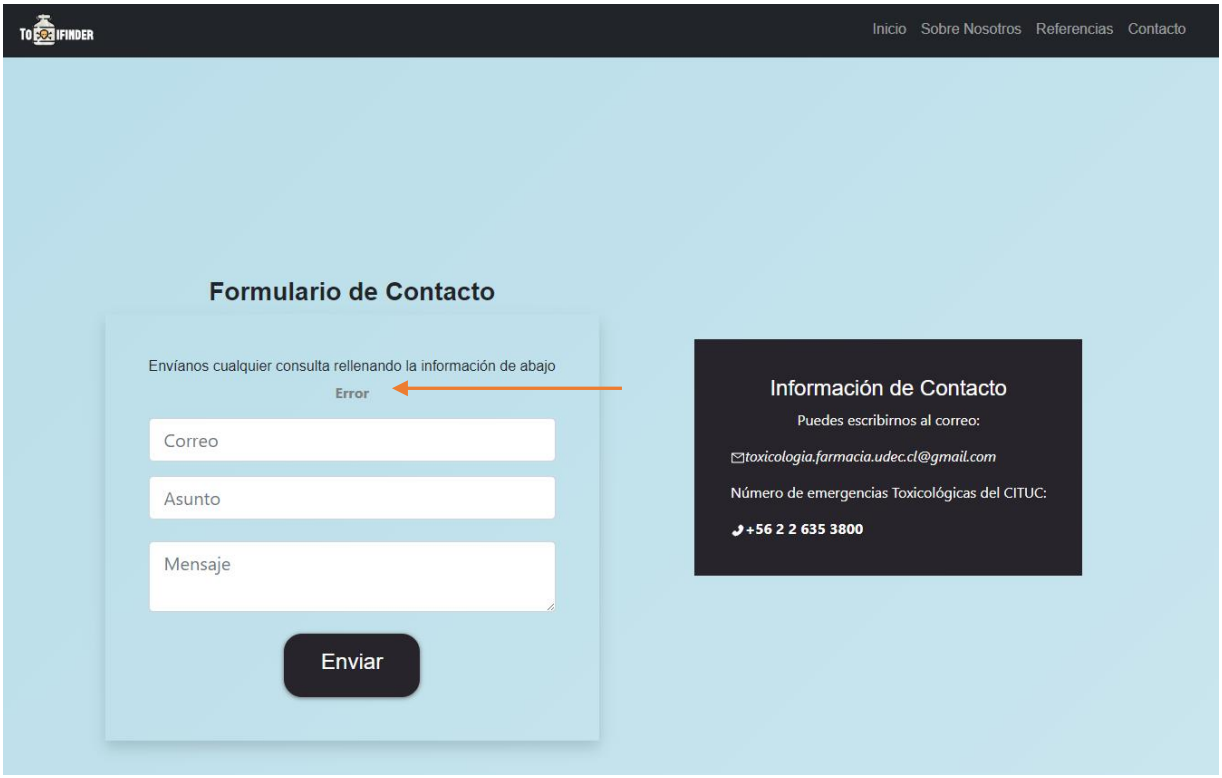

Figura 22. Vista de la página de contacto con un error en el formulario.

B. Formulario de contacto

A modo de comprobar el funcionamiento del servicio de email se realizó una prueba del formulario de contacto ingresando algunos datos. Si vemos la Figura 23, se puede apreciar que el servidor envía correctamente el mensaje de estado que confirma la conexión de Mongoose con Mailtrap.

Por lo tanto, nodemailer en el lado del servidor y el formulario creado en el lado del cliente, se encuentran correctamente configurados y también contectados entre sí intercambiando información.

Figura 23. Correo enviado con éxito desde el Formulario de Contacto.

Para confirmar que el correo llegó con éxito, se ingresó a la bandeja de entrada del correo, donde se apreció el mensaje respetando el diseño definido en la programación y con los campos correctamente llenados (ver Figura 24).

Figura 24. Ejemplo de correo de consulta mediante Formulario de Contacto.

5.2.3 Página Sobre Nosotros

En esta página el componente más importante son las Cards del equipo encargado el proyecto. En la programación se buscó generar cuadros informativos que resumieran la participación de cada uno de los miembros del equipo (ver Figura 25).

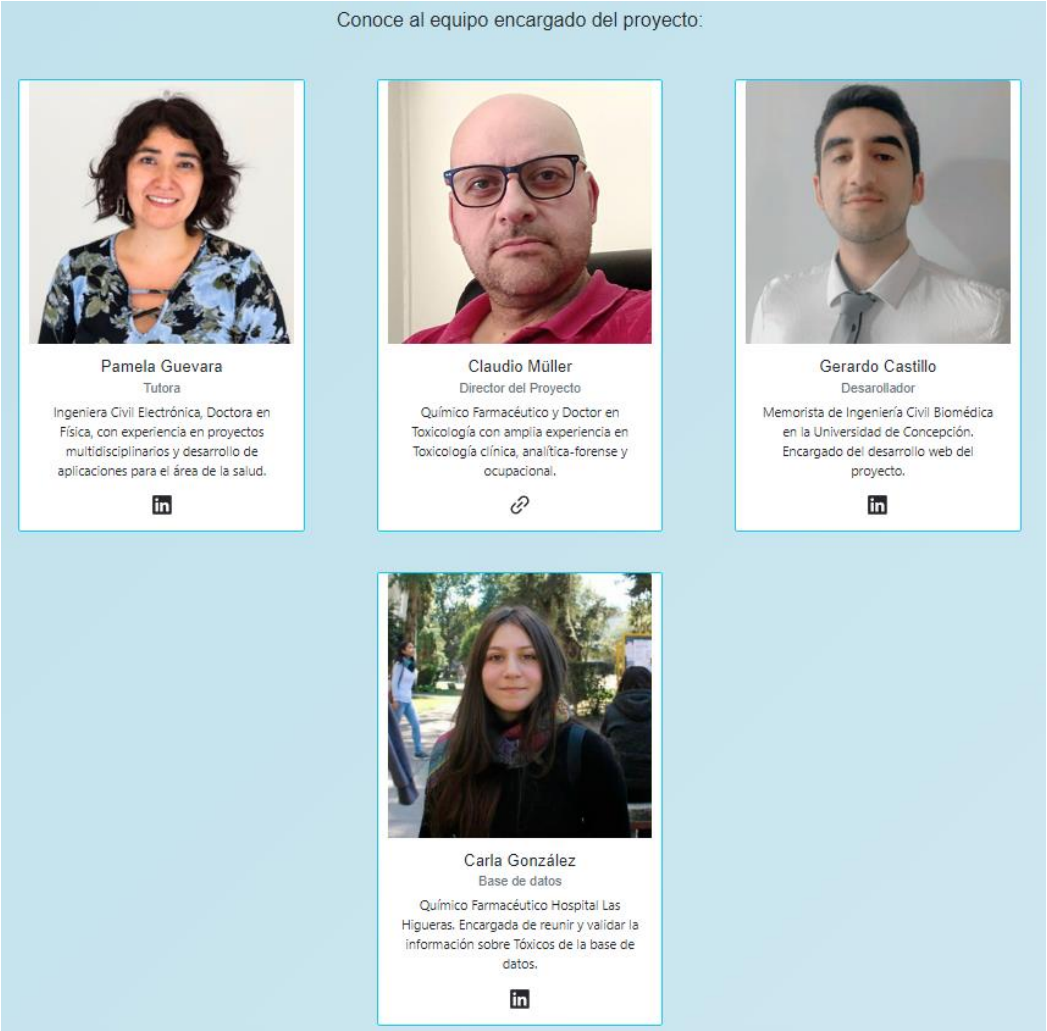

Figura 25. Vista de las tarjetas de información de la página Sobre Nosotros.

Además, se consiguió agregar un nuevo hero para esta página con la misión y el objetivo del proyecto. Finalmente se sumaron los logos de las facultades participantes al final de la página.

5.2.4 Página de Referencias

La página de referencia se mostró correctamente con toda la información ordenada por categoría, utilizando la función ListGroup (ver Figura 26).

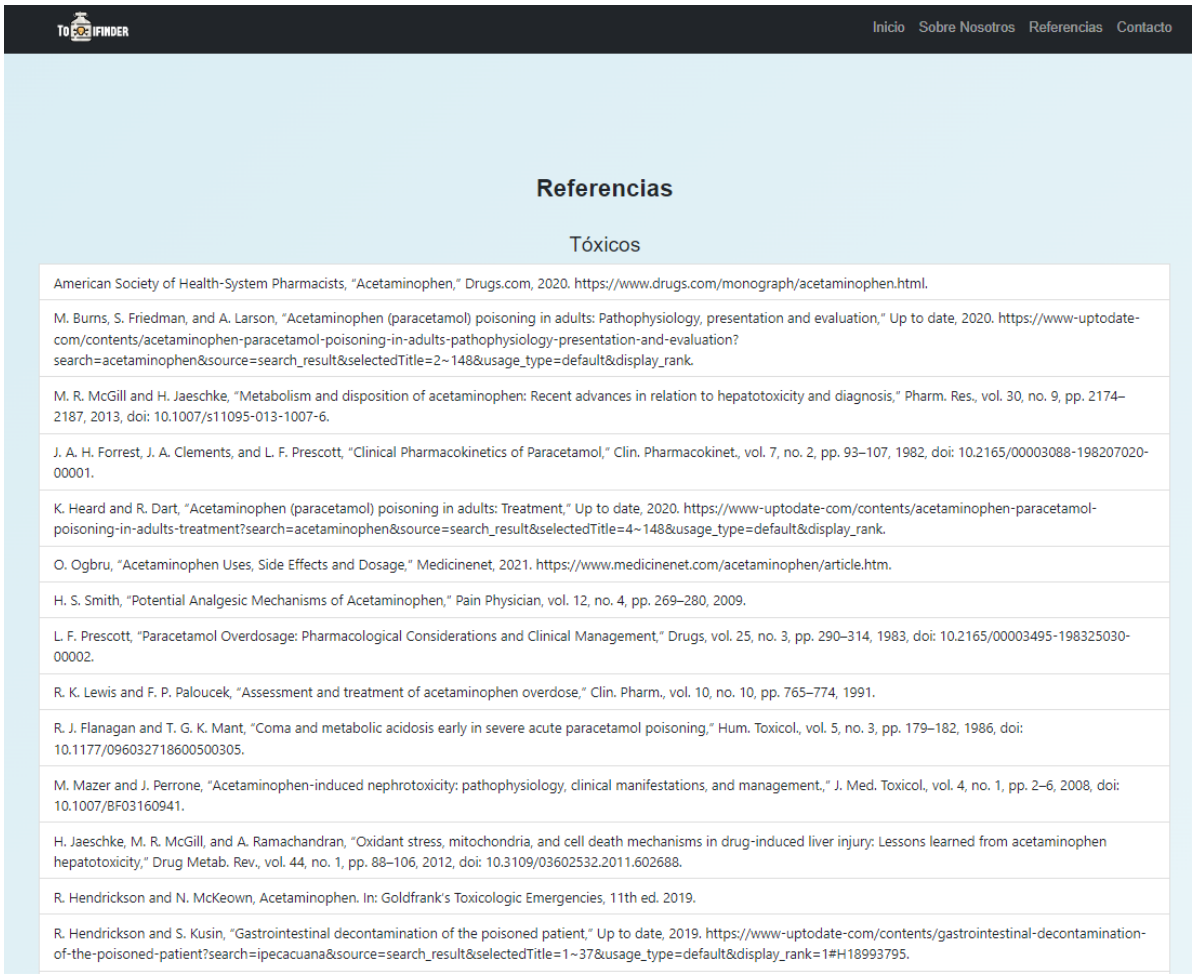

Figura 26. Vista de la página de referencias.

5.3. Compatibilidad con varias resoluciones de pantalla

Uno de los principios que ofrecen las aplicaciones web desarrolladas con tecnología MERN es que poseen una compatibilidad para distintas resoluciones de pantalla con la característica “Responsive Web Design” o “Diseño de web adaptable”. Es por ello que en la programación se configuraron los cambios visuales del Frontend a distintas resoluciones, como es el caso del Navbar o barra de navegación en el header de la Web App que se acopla en dispositivos de resoluciones más pequeñas.

Para probar el comportamiento de la Web App se consideró el modelo Samsung Galaxy S8+, un iPad Air y un PC con Microsoft Edge (ver Tabla 6).

Dispositivo	Sistema Operativo	Resolución de pantalla	Tamaño de dispositivo
Samsung Galaxy S8+	Android 7.0	360x740	Pequeño
iPad Air 4	iPadOs 14	820x1180	Mediano
Computador 16:9	Windows 11	1920x1080	Grande

Tabla 6. Dispositivos de prueba de pantalla.

Se utilizaron las herramientas de desarrollador de Microsoft Edge para simular los dispositivos Windows 11. Se pudo ver que la aplicación web responde correctamente a cualquier cambio de resolución según los ajustes establecidos en la programación, Por ello es correcto afirmar que es compatible con distintos tipos de resoluciones de dispositivos, ya sea celulares, tablets o computadores. Se puede apreciar de las Figura 27, Figura 28 y Figura 29 como la aplicación reacciona ante los cambios de resolución.

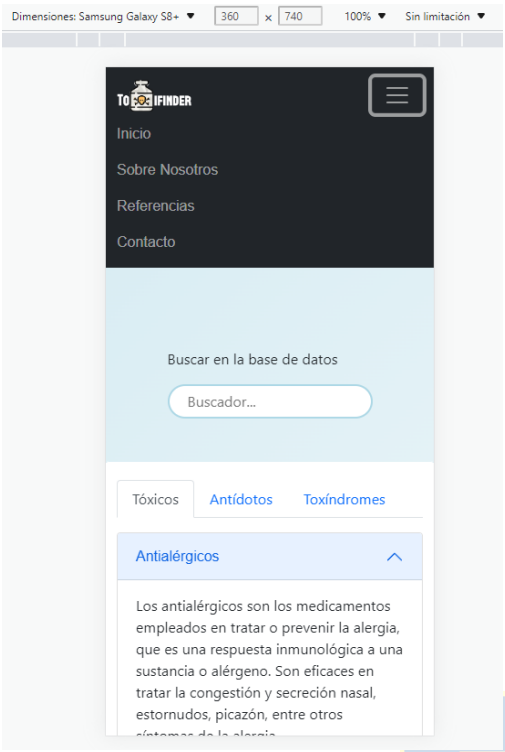

Figura 27. Vista desde Samsung Galaxy S8+

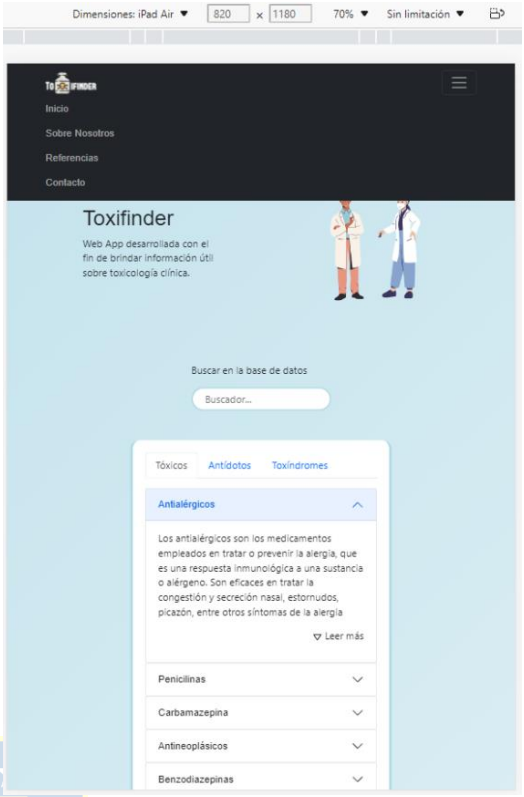

Figura 28. Vista desde iPad Air.

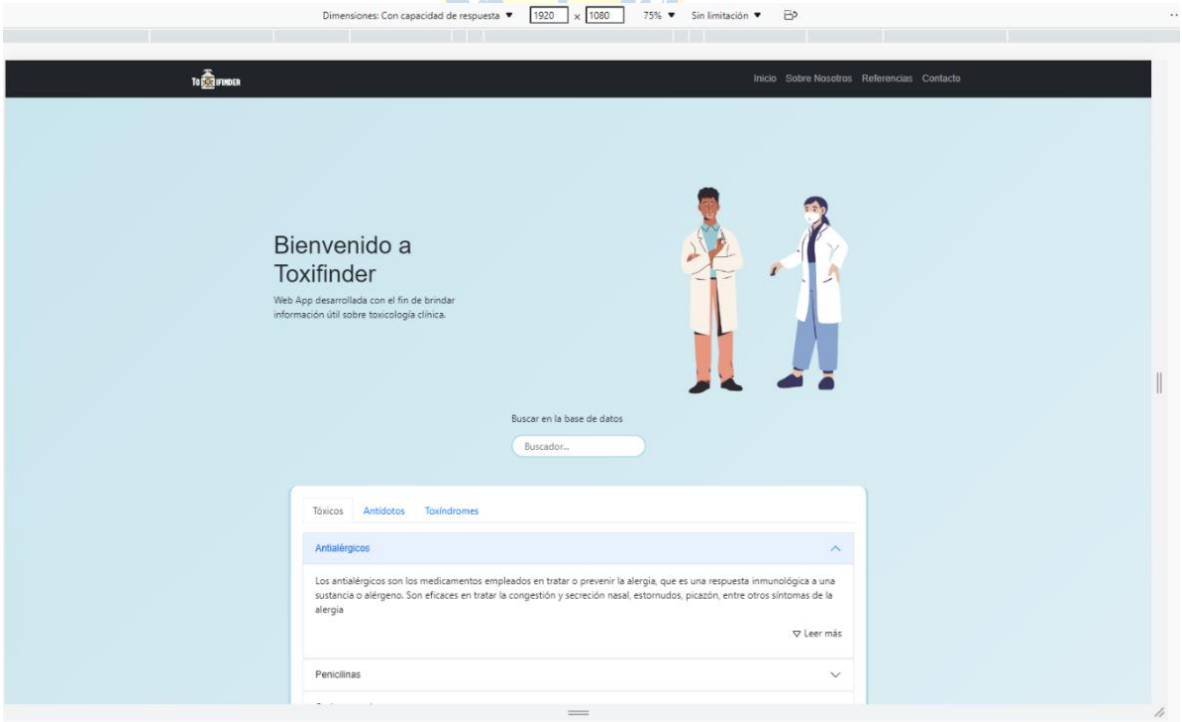

Figura 29. Vista desde computador 16:9

5.4. Encuesta a profesionales

Se realizó una encuesta a tres profesionales relacionados al área clínica para evaluar tanto el rendimiento como la presentación de la Web App.

Primeramente, se les preguntó por su tipo de conexión y la calidad de esta. Se obtuvo que la conexión inalámbrica es la más utilizada con dos votos y las redes móviles en segundo lugar con solo un voto.

Se consultó por la presentación en general, donde tres profesionales lo evaluaron con la nota máxima (Figura 30). Luego, se preguntó si su uso es intuitivo al usuario, consiguiendo los mismos resultados anteriores (Figura 31).

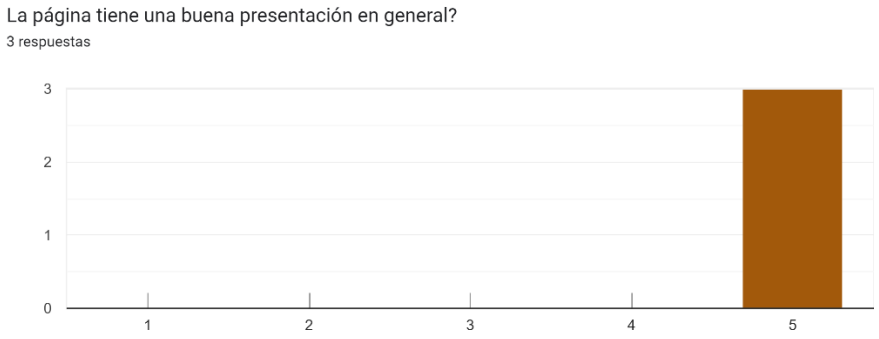

Figura 30. Gráfico de resultados a pregunta de presentación a profesionales.

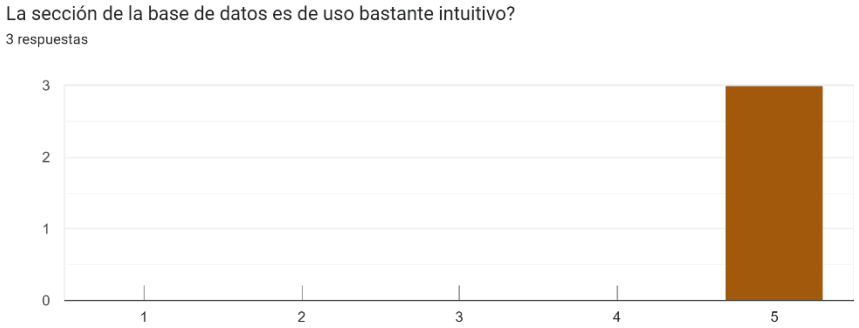

Figura 31. Gráfico de resultados a pregunta de uso intuitivo a profesionales.

Se le pidió a los encuestados que realizaran una búsqueda específica en la base de datos con la palabra “Paracetamol”, de esta manera revisar la facilidad de dar con el resultado y la velocidad de respuesta de este. Se consiguió que los tres profesionales evaluaran muy positivamente la búsqueda de este medicamento y que calificaran como “excelente” la velocidad de respuesta (ver Figura 32).

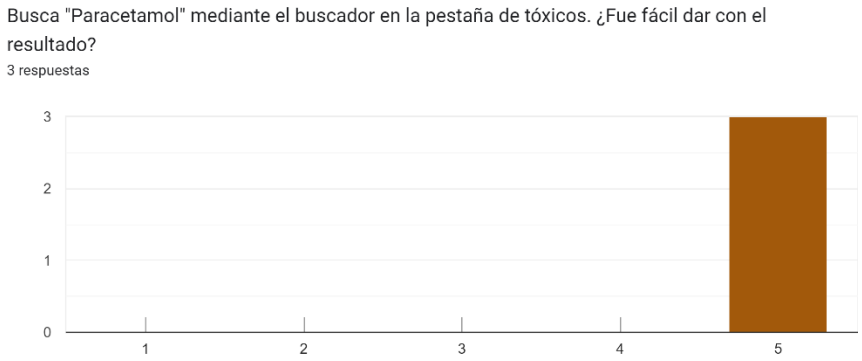

Figura 32. Gráfico de resultados a pregunta de búsqueda de Paracetamol en la base de datos.

El 100% de los encuestados declaró que usaría la Web App como medio de información sobre tóxicos y a su vez definieron que la información se presenta de forma breve y concisa.

Mediante un recuadro de comentarios se obtuvo una realimentación, en donde, uno de los interesados mencionó que “se podrían agregar más elementos para hacer más atractiva la aplicación como una sección de noticias y un elemento de muestra de fiabilidad del sitio”. El otro comentario manifiesta que “se deben modificar ciertas abreviaturas o sinónimos en la información de la base de datos, por ejemplo Pb, Hg o acetaminofeno”.

Discusión y conclusiones

6.1. Discusión

El correcto diseño supuso una de las partes más importantes para el proyecto, ya que, optimizó bastante el tiempo y ordenó todo el cimiento de la programación. Se consiguió esquematizar la aplicación a nivel general para seccionar la correspondencia de las bibliotecas con los respectivos archivos javascript que las utilizarían.

Se crearon todas las partes que se encuentran fuera de la programación antes de la misma, para que al momento de requerirlas estas ya estuvieran funcionando. Se optó por utilizar PaaS en vez de la máquina local para poder requerir realimentación mediante la encuesta realizada y además para que se pueda probar desde otros dispositivos, ya sea por los tutores u otros docentes.

La programación del lado del servidor se destacó por ser bastante breve y concisa con respecto a lo que se esperaba del lado del cliente. El mayor problema radicó en el servicio de mensajería, debido a los estrictos permisos de seguridad que se requieren para utilizar servicios masivos como Gmail u Outlook. Por esta razón se utilizó mailtrap a modo de realizar todas las pruebas del apartado de formulario de contacto.

Para el lado del cliente, es importante destacar que React-Boostrap facilita todo el tema estético en cuanto a todos los elementos renderizados por su biblioteca. Sin embargo, se presentaron muchos inconvenientes de orden visual que fueron solucionados con configuraciones básicas de css. Se puede volver a rescatar la versatilidad de React por minimizar el trabajo de programación a la hora de generar elementos y como estos se escalan de manera automática cuando se cambia la resolución. Cabe mencionar el rol que tuvieron los foros de preguntas de programación como overflow, ya que, en estos se extrajeron muchos ejemplos de referencia para solucionar errores de código y mejorar su funcionamiento.

Se pudo notar mediante la prueba de compatibilidad de varias resoluciones de pantalla que la aplicación web cumplió con todas las expectativas en cuanto a su responsividad respecto a la resolución del dispositivo que se utilice, logrando así la característica de compatibilidad con múltiples tipos de dispositivos. Cabe señalar que el público objetivo de este proyecto utiliza desde celulares hasta notebooks de variados modelos.

6.2. Conclusión final

De acuerdo a toda la evidencia y las pruebas realizadas anteriormente, se demuestra que este proyecto logró cumplir con éxito los objetivos planteados desde un comienzo.

Se había comprometido "Diseñar la web app y base de datos, considerando contenidos a desplegar y las funcionalidades por ofrecer". Para cumplir con dicho objetivo se estudiaron las posibles variantes y se optó utilizar tecnologías más actuales, las cuales facilitaron el diseño debido a la gran existencia de ejemplos disponibles, por lo que no fue un estancamiento a la hora de diseñar.

Se necesitaba enfocar la web app en una aplicación de fácil uso e intuitiva, lo cual se logró integrando el buscador de la base de datos bien resaltado en la página de inicio y, utilizando React Bootstrap se implementó toda la componente que muestra los resultados de la db. Los encuestados manifestaron estar de acuerdo con la facilidad de utilizar esta sección y consideraron que presenta una buena apariencia.

Para comprobar todas las funcionalidades y la optimización de la aplicación se realizaron distintas pruebas, lo que corrobora el correcto funcionamiento y concluyendo así con una web app responsiva, rápida e intuitiva.

6.3. Trabajo futuro

Los resultados del proyecto realizado crean la base para una futura aplicación web de información toxicológica que permita a estudiantes y a profesionales adquirir información de vital importancia a la hora de desempeñar sus labores. No se descarta la posibilidad de unificar toda la información disponible sobre toxicología clínica y así complementar la base de datos actual, ya que, cabe recordar que solo contiene el 10% de los tóxicos más importantes registrados.

Por otra parte, los códigos de programación de este proyecto pueden ser mejorados para implementar nuevas funcionalidades y/o mejorar la interfaz de la web app. Se abre la posibilidad de cambiar el servicio de mensajería utilizado por uno dedicado a la aplicación web. Asimismo, se abre la posibilidad de agregar la sección de noticias que recomendó uno de los encuestados.

Con mayor altura de miras se podría trabajar en conjunto al CITUC para generar una nueva versión apuntada a un mayor público y con mejores prestaciones.

Bibliografía

[1] Carla González, “Elaboración De Las Bases Para El Desarrollo De Una Aplicación Para Dispositivos Móviles De Ayuda En Toxicología Clínica”, Facultad de Química y Farmacia, Universidad de Concepción, 2020.

[2] Ricardo León Aceitón y Sebastián Meza Muñoz, “Brecha en el uso de internet”. [Online]. Available: https://paisdigital.org/brecha-en-el-uso-de-internet-2020/. [Consultado: abr. 11, 2022]

[3] Katie Lawson, “What is a single page application”. [Online]. Available: https://www.bloomreach.com/en/blog/2018/07/what-is-a-single-page-application.html. [Consultado: abr. 11, 2022]

[4] “Qué es una web app y qué clases hay?” [Online]. Available: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-una-web-app-y-que-clases-hay/. [Consultado: abr. 11, 2022]

[5] “Toxicology Mentoring and Skills Development Training”. [Online]. Available: https://www.toxmsdt.com/0-toxtutor-home.html. [Consultado: abr. 14, 2022]

[6] “Diccionario Médico CUN España”. [Online]. Available: https://www.cun.es/diccionario-medico. [Consultado: abr. 14, 2022]

[7] “Historia: Renacimiento, Paracelso”. [Online]. Available: https://www.ugr.es/~ajerez/proyecto/t1_3.htm. [Consultado: abr. 14, 2022]

[8] “SpotlightMed: MedCoach”. [Online]. Available: https://spotlightmed.com/. [Consultado: abr. 14, 2022]

[9] “Toxicología Hoy Información de consultas toxicológicas”. [Online]. Available: https://toxicologiahoy.com. [Consultado: abr. 14, 2022]

[10] “Google Playstore: Toxicología Hoy”. [Online]. Available: https://play.google.com/store/apps/details?id=com.toxicologiahoy.app&hl=es_CL&gl=US. [Consultado: abr. 14, 2022]

[11] “Google Playstore: iTox”. [Online]. Available: https://play.google.com/store/apps/details?id=org.adaliafarma.itox. [Consultado: abr. 14, 2022]

[12] “ToxicologíaNet: Programa de información y formación en Toxicología Clínica”. [Online]. Available: https://www.fetoc.es/toxicologianet/index.htm. [Consultado: abr. 14, 2022]

[13] “Single page app vs multi page app”. [Online]. Available: https://lvivity.com/single-page-app-vs-multi-page-app. [Consultado: abr. 15, 2022]

[14] Hiren Dhaduk, “An Ultimate Guide to Web Application Architecture”, 2021. [Online]. Available: https://www.simform.com/blog/web-application-architecture/. [Consultado: abr. 15, 2022]

[15] Francisco Palomares, “¿Qué es una web SPA? - Single Page Application”. [Online]. Available: https://www.kikopalomares.com/blog/que-es-una-web-spa-single-page-application. [Consultado: abr. 15, 2022]

[16] Anastasia Zakrevskaya, “What’s the Difference Between Single-Page and Multi-Page Apps”, 2018. [Online]. Available: https://rubygarage.org/blog/single-page-app-vs-multi-page-app. [Consultado: abr. 15, 2022]

[17] Maksym Churylov, “SPA vs MPA: The definitive guide for decision makers”. [Online]. Available: https://www.mindk.com/blog/single-page-applications-the-definitive-guide/. [Consultado: jun. 13, 2022]

[18] Nicole Chapaval, “Que es frontend y backend”, 2018. [Online]. Available: https://platzi.com/blog/que-es-frontend-y-backend/. [Consultado: abr. 15, 2022]
[19] “Qué es el Frontend?” [Online]. Available: https://www.suratica.es/que-es-el-frontend/. [Consultado: may 12, 2022]
[20] “NOSQL vs SQL. Key differences and when to choose each”. [Online]. Available: https://pandorafms.com/blog/nosql-vs-sql-key-differences/. [Consultado: may 12, 2022]
[21] Natalia Campana, “Qué hace un administrador de bases de datos?”, 2020. [Online]. Available: https://www.freelancermap.com/blog/es/que-hace-administrador-bases-datos/. [Consultado: abr. 17, 2022]
[22] “Administración de bases de datos”. [Online]. Available: https://saludelectronica.com/administracion-de-bases-de-datos/. [Consultado: abr. 17, 2022]
[23] Gabriela Muente, “Guía completa del Framework: qué es, cuáles tipos existen y por qué es importante en Internet”, 2020. [Online]. Available: https://rockcontent.com/es/blog/framework/. [Consultado: abr. 17, 2022]
[24] “10 Best free web application Frameworks”. [Online]. Available: https://www.linuxlinks.com/applicationframeworks/. [Consultado: abr. 18, 2022]
[25] “Web frameworks rankings”. [Online]. Available: https://hotframeworks.com/. [Consultado: abr. 19, 2022]
[26] Miguel Parada, “MERN Stack: Qué es y qué ventajas ofrece”, 2020. [Online]. Available: https://openwebinars.net/blog/mern-stack-que-es-y-que-ventajas-ofrece/. [Consultado: abr. 25, 2022]
[27] “How to Use MERN Stack: A Complete Guide”. [Online]. Available: https://www.mongodb.com/languages/mern-stack-tutorial. [Consultado: abr. 25, 2022]
[28] H. A. Quintana-Cruz, “Implementación de aplicaciones isomórficas con Javascript”, *Interfases*, vol. 0, núm. 008, p. 143, abr. 2015, doi: 10.26439/interfases2015.n008.580.
[29] “Mongoose v6.5.2 Documentation”. [Online]. Available: https://mongoosejs.com/docs/connections.html. [Consultado: may 22, 2022]
[30] “SMTP transport by Nodemailer”. [Online]. Available: https://nodemailer.com/smtp/. [Consultado: may 24, 2022]
[31] “Express Routing Documentation”. [Online]. Available: http://expressjs.com/en/guide/routing.html. [Consultado: may 12, 2022]
[32] “How Heroku Works”. [Online]. Available: https://devcenter.heroku.com/articles/how-heroku-works. [Consultado: may 12, 2022]
[33] “POST Requests - Axios Docs”. [Online]. Available: https://axios-http.com/docs/post_example. [Consultado: may 12, 2022]
[34] “React-Bootstrap Documentation”. [Online]. Available: https://react-bootstrap.github.io/getting-started/introduction. [Consultado: jun. 02, 2022]
[35] Diego Zoracky, “Convert excel to json NPM JS”. [Online]. Available: https://www.npmjs.com/package/convert-excel-to-json. [Consultado: may 09, 2022]
[36] “MongoDB: Get Started with Atlas”. [Online]. Available: https://www.mongodb.com/docs/atlas/getting-started/. [Consultado: may 12, 2022]
[37] Jessica Clark, “Netlify vs Heroku | ¿Cuáles son las diferencias?” [Online]. Available: https://blog.back4app.com/es/netlify-vs-heroku-cuales-son-las-diferencias/#Que_es_Netlify. [Consultado: may 12, 2022]
[38] “Welcome to Netlify Docs”. [Online]. Available: https://docs.netlify.com/. [Consultado: may 12, 2022]

[39] Hal Friday, “Creating Models for MongoDB with Mongoose”. [Online]. Available: https://dev.to/halented/part-2-creating-models-for-mongodb-with-mongoose-575d. [Consultado: may 22, 2022]
[40] “What is process.env.PORT in Node.js?” [Online]. Available: https://stackoverflow.com/questions/18864677/what-is-process-env-port-in-node-js. [Consultado: may 22, 2022]
[41] “Express.js | app.listen() Function”. [Online]. Available: https://www.geeksforgeeks.org/express-js-app-listen-function/. [Consultado: may 22, 2022]
[42] “Express cors: Connect/Express middleware”. [Online]. Available: https://expressjs.com/en/resources/middleware/cors.html. [Consultado: may 22, 2022]
[43] “express Tutorial Multiple Routes”. [Online]. Available: https://riptutorial.com/express/example/16315/multiple-routes. [Consultado: may 22, 2022]
[44] “dotenv npm Documentation”. [Online]. Available: https://www.npmjs.com/package/dotenv. [Consultado: may 24, 2022]
[45] “Express.js json function”. [Online]. Available: https://www.geeksforgeeks.org/express-js-express-json-function/. [Consultado: may 24, 2022]
[46] “Getting started guide - Mailtrap.io”. [Online]. Available: https://help.mailtrap.io/article/12-getting-started-guide. [Consultado: may 26, 2022]
[47] “Message configuration by Nodemailer”. [Online]. Available: https://nodemailer.com/message/. [Consultado: may 24, 2022]
[48] “React Router: Declarative Routing for React.js”. [Online]. Available: https://v5.reactrouter.com/web/api/BrowserRouter. [Consultado: jun. 02, 2022]
[49] Anthony Malla, “Todas las partes de una página web: Estructura y contenido”. [Online]. Available: https://pleybast.com/sitio-web/partes-de-una-pagina-web-estructura-y-contenido/. [Consultado: jul. 28, 2022]
[50] “React Icons Library”. [Online]. Available: https://react-icons.github.io/react-icons/. [Consultado: jun. 12, 2022]
[51] “React Bootstrap Layout Grid Documentation”. [Online]. Available: https://react-bootstrap.github.io/layout/grid/. [Consultado: jun. 12, 2022]
[52] “Data Fetching con React Usando useState y useEffect”. [Online]. Available: https://www.escuelafrontend.com/articulos/data-fetching-con-react. [Consultado: jun. 12, 2022]
[53] “Tabbed components on React-Boostrap”. [Online]. Available: https://react-bootstrap.github.io/components/tabs/. [Consultado: jun. 12, 2022]
[54] Janith Kasun, “Making Asynchronous HTTP Requests in JavaScript with Axios”. [Online]. Available: https://stackabuse.com/making-asynchronous-http-requests-in-javascript-with-axios/. [Consultado: jun. 16, 2022]
[55] “Initializing and using `sessionStorage` in React”. [Online]. Available: https://stackoverflow.com/questions/40399873/initializing-and-using-sessionstorage-in-react. [Consultado: jun. 20, 2022]
[56] “HTML Styles CSS”. [Online]. Available: https://www.w3schools.com/html/html_css.asp. [Consultado: jun. 20, 2022]
[57] “HTML Favicon W3 School Docs”. [Online]. Available: https://www.w3schools.com/html/html_favicon.asp. [Consultado: jun. 20, 2022]
[58] “Catch all redirect for react app in netlify - Stack overflow”. [Online]. Available: https://stackoverflow.com/questions/55990467/catch-all-redirect-for-create-react-app-in-netlify. [Consultado: abr. 14, 2022]

[59] “What is Deployment in software and web development?” [Online]. Available: https://umbraco.com/knowledge-base/deployment/. [Consultado: jun. 20, 2022]
[60] “What is a build in programming context?” [Online]. Available: https://www.techtarget.com/searchsoftwarequality/definition/build. [Consultado: jun. 20, 2022]
[61] “Heroku Dev Center: Deploying with Git”. [Online]. Available: https://devcenter.heroku.com/articles/git. [Consultado: jun. 20, 2022]
[62] Eli Williamson y Jason Lengstorf, “A Step-by-Step Guide: Deploying on Netlify”. [Online]. Available: https://www.netlify.com/blog/2016/09/29/a-step-by-step-guide-deploying-on-netlify/. [Consultado: jun. 20, 2022]
[63] “Inspect network activity on Microsoft Edge”. [Online]. Available: https://docs.microsoft.com/en-us/microsoft-edge/devtools-guide-chromium/network/. [Consultado: jun. 20, 2022]

Anexo A. Programación detallada de la web app

1.1. Introducción

Es bastante importante entender el funcionamiento del código para saber cómo abordarlo a la hora de realizar mejoras. En el presente anexo se datan segmentos de la programación con mayor profundidad tanto del lado del servidor como del lado del cliente.

1.2. Creación de modelos para la base de datos

Se crea un esquema para antídotos, tóxicos y toxíndromes, donde cada uno de los campos que contendrá información son del tipo String. Para realizar esta función se utiliza “mongoose.Schema” (ver Figura 1).

```
const antidotosSchema = new mongoose.Schema({
  name: String,
  description: String,
  mechanism: String,
  indications: String,
  doses: String,
  counterindic: String,
  effects: String
});
```

Figura 1. Ejemplo de esquema para antídotos.

Se crea cada modelo con su respectivo esquema utilizando la función “mongoose.model” (Figura 2) con el fin de poder exportar las constantes para posteriormente ser leídas en el enrutador del servidor. Se usó “module.exports” para retornar las constantes hacia archivos externos del models.js

```
const antidotosModel = mongoose.model('Antidoto', antidotosSchema);
const toxicosModel = mongoose.model('Toxico', toxicosSchema);
const toxindromesModel = mongoose.model('Toxindrome', toxindromesSchema);
```

Figura 2. Asignación de los esquemas a cada modelo de la base de datos.

1.3. Enrutador del servidor

El direccionamiento del Backend es facilitado por la biblioteca de Express JS mediante el archivo “Routes.js”. Este direccionamiento permite generar las solicitudes de GET, SEND y FIND. Se crea un objeto denominado homeRouter el cual contendrá todas las rutas de las colecciones. Esto quiere decir que en la API podemos entrar a cada colección de la base de datos mediante esta conexión. Con “homeRouter.get” en la ruta de antídotos, tóxicos y toxíndromes se llaman los modelos creados en el archivo anterior, para así adjuntar la información en ellos. Mediante la función SEND en el objeto de respuesta declarado como “res” se adjuntan los datos de “docs”, parámetro que entraña los documentos de cada colección de la DB (Figura 3). Se finalizó con “module.exports” para extraer este nuevo objeto “homeRouter”.

```
// Models
const {antidotosModel, toxicosModel, toxindromesModel}  = require('../Model/Model')

homeRouter.get("/antidotos",(req, res, next)=>{
    antidotosModel.find((err, docs)=>{
      res.send(docs)
    })
  })
homeRouter.get("/toxicos",(req, res, next)=>{
    toxicosModel.find((err, docs)=>{
      res.send(docs)
    })
  })
homeRouter.get("/toxindromes",(req, res, next)=>{
    toxindromesModel.find((err, docs)=>{
      res.send(docs)
    })
  })
```

Figura 3. Direccionamiento de la base de datos.

1.4. Programación de la aplicación para el servidor

Se definió la constante del puerto con “port = process.env.PORT || 5000”, esto quiere decir que el puerto será cualquiera que el entorno defina o el puerto 5000, un puerto poco utilizado en Windows. Se realizó esta manipulación del puerto porque el entorno estará definido por Heroku, plataforma como servicio en la nube que seleccionará automáticamente un puerto disponible en su hosting al ejecutar nuestro servidor. Por otra parte, si se corre el servidor en un entorno local, utilizará el puerto 5000 obligatoriamente.

Con la función “dotenv.config()” se carga el archivo “.env” situado en la carpeta raíz del servidor. Fichero que tiene como función alojar datos de manera oculta del navegador. En este caso la información sensible es el enlace directo a la base de datos que incluye la llave a la misma bajo la constante “CONNECTION_URL” (ver Figura 4).

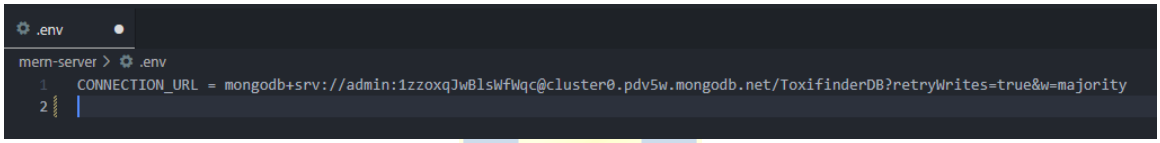

Figura 4. Archivo “.env”

La conexión se realiza con el uso de “mongoose.connect”, empleando la llave comentada anteriormente. Para confirmar la conexión se envía un log hacia la consola con un mensaje de éxito o la causa del error (Figura 5). Se maneja el control de acceso con “CORS”, mecanismo para la obtención de permiso para acceder a recursos seleccionados desde un servidor, en un distinto dominio al que pertenece (Figura 6). Se crea la aplicación de Express JS llamada “app” y se enlaza el middleware (lógica de intercambio de información entre aplicaciones) a una instancia del objeto de aplicación utilizando la función “app.use”. Realizando el paso anterior se asigna “homeRouter” a la página raíz del servidor (Figura 6). Luego, la función app.listen() se usa para vincular y escuchar las conexiones en el host y puerto definido (Figura 6).

```
mongoose.connect(process.env.CONNECTION_URL, {
  serverSelectionTimeoutMS: 5000,
  useNewUrlParser: true,
  useUnifiedTopology: true
  }).then(db => console.log('Base de datos conectada con éxito.'))
  .catch(err => console.log(err.reason));
```

Figura 5. Conexión a base de datos.

```
app.use(cors())
// Infraestructura web de ruteo y middleware
app.use('/', homeRouter)

// Llamamos el puerto
app.listen(port, ()=>{
  console.log("Server ON en el puerto ", port)
})
```

Figura 6. CORS, homeRouter y llamado al puerto del servidor.

Se configuran las dos variables principales de nodemailer (encargado de enviar correos) llamadas “transporter” y “mailOptions”. El transportador tiene como función principal establecer el servicio de mensajería, vale decir, su host, el puerto y la autenticación (nombre de usuario y su contraseña). Las opciones definen destinario, remitente, asunto, contenido y formato del mensaje, este último en código html. Para efectos de prueba se estableció un servicio de mensajería provisorio suministrado por Mailtrap, una PaaS que de cierta manera “atrapa” los correos que utilizan su servidor SMTP. El transportador envía el correo con la función “sendMail” de nodemailer. Para confirmar el estado del envío de este correo se envía un mensaje hacia el cliente almacenado en “respMesg” siguiendo el mismo concepto de documento json (Figura 7).

```
transporter.sendMail(mailOptions, function(error, info){
    if (error)
    {
      console.log('Error en formulario');
      res.json({status: true, respMesg: 'Ha ocurrido un error'})
    }
    else
    {
      console.log('Correo enviado:' + info.response);
      res.json({status: true, respMesg: 'Correo enviado con éxito'})
    }

  });
})
```

Figura 7. Transportador: Envío de correo y comprobación de errores.

1.5. Programación de la aplicación principal del Frontend

Mediante React-router-dom se utiliza la función BrowserRouter para generar el direccionamiento del lado del cliente, donde se le asigna una ruta a cada página de la aplicación. Se deja la ruta raíz para la página de inicio.

En la Figura 8 se visualiza la función principal App con las direcciones o rutas del cliente identificadas con el comando path. A estas rutas se les asigna su respectiva página.

```
function App() {
  return (
    <BrowserRouter >
      <Routes>
        <Route path='/about' element={<About />} />
        <Route path='/contact' element={<Contact />} />
        <Route path='/' element={<Home />} />
        <Route path='/refs' element={<Refs/>} />
      </Routes>
    </BrowserRouter>
  );
}
```

Figura 8. Función principal App Frontend.

1.6. Componentes del Frontend

A continuación, se detalla la programación algunas componentes utilizadas en las páginas del Frontend.

- Header: Consiste en una barra de navegación de la biblioteca de React con todos los enlaces de la web app. Mediante la función Navbar se creó la barra y se le establecieron parámetros para dejarla siempre fija en la parte superior y también que se colapse en dispositivos grandes (≥1200 px). Se ajustó una imagen de "branding" con el logotipo de la aplicación (Toxifinder) en 35 px de alto. Utilizando "Nav.Link" se definen todos los enlaces del Header (ver Figura 9).

```
export default function Header() {
  return <div>
        <Navbar fixed="top" collapseOnSelect expand="lg" bg="dark" variant="dark">
          <Container>
          <Navbar.Brand href="/"><img style={{height:'35px'}} src={Logo} alt=""/></Navbar.Brand>
          <Navbar.Toggle aria-controls="responsive-navbar-nav" />
          <Navbar.Collapse id="responsive-navbar-nav">
            <Nav className="me-auto">
            </Nav>
            <Nav className="fuente">
              <Nav.Link href="/">Inicio</Nav.Link>
              <Nav.Link href="/about">Sobre Nosotros</Nav.Link>
              <Nav.Link href="/refs">Referencias</Nav.Link>
              <Nav.Link href="/contact">Contacto</Nav.Link>
            </Nav>
          </Navbar.Collapse>
          </Container>
        </Navbar>
  </div>
}
```

Figura 9. Código de la barra de navegación en la función principal del Header.

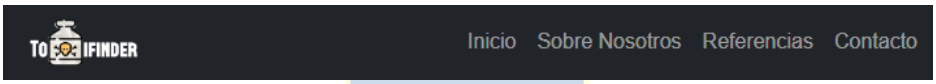

Figura 10. Vista de la cabecera, se aprecian los enlaces de la Web.

- Hero: Es la sección que primero ve el usuario al entrar en un sitio web. Se dividió en un Hero para poner el mensaje de bienvenida a la web (Figura 11) y en otro para visualizar información sobre la base de datos (Figura 12). Se utilizó un Container (especie de rejilla para dividir los espacios a utilizar) de una fila y dos columnas para ambos componentes.

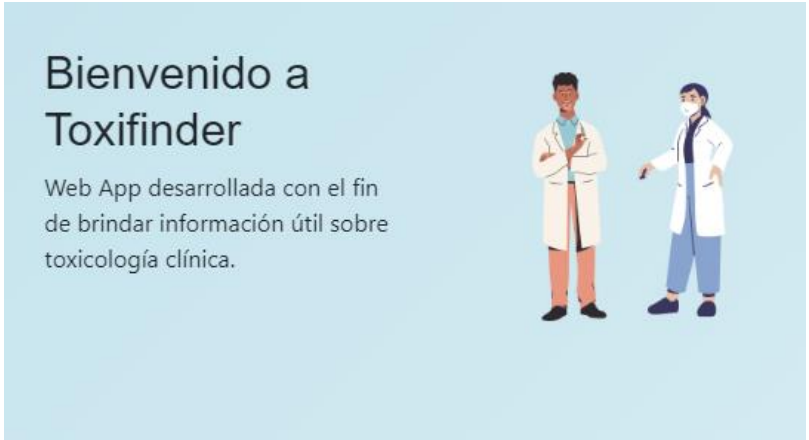

Figura 11. Vista del primer Hero, se aprecia el mensaje de Bienvenida.

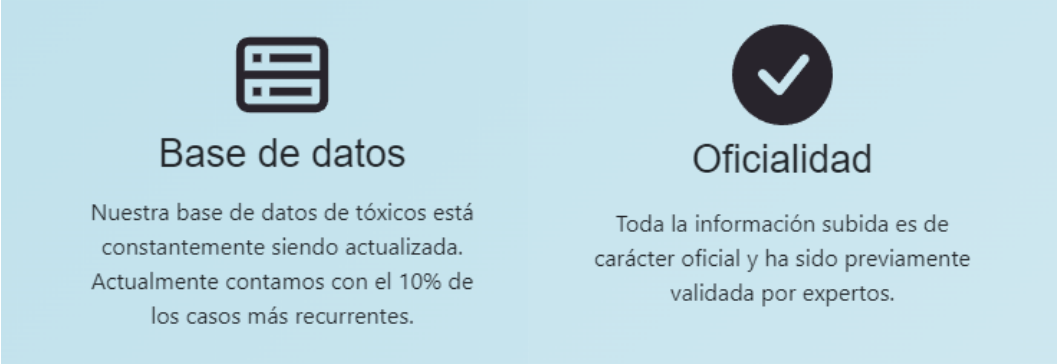

Figura 12. Vista del segundo Hero, se denota información de la base de datos.

- DBSearch: Componente más importante para la base de datos que contiene el buscador y la información que se comunica desde el Backend. Aquí se involucraron tres archivos JS para representar los datos de cada categoría a utilizar (Tóxicos, Antídotos y Toxíndromes).

  Primeramente, se declaró el enlace de nuestro Backend (dirección de Heroku). Para realizar la lectura de datos desde nuestro servidor se utilizó “useEffect”, función que permite hacer el “fetch” desde esta dirección externa. Cuando se realiza el fetch se almacenan las variables de estado en cada una de las categorías mencionadas como información json (ver Figura 13).

```
const url = 'https://toxifinder.herokuapp.com/' //Dirección del Backend
// const url = 'http://192.168.1.111:5000/' // Local IP:PORT
useEffect(() => {
    fetch(url + 'toxicos')
        .then(data => data.json())
        .then(data => {
            setToxicos(data)
        })

    fetch(url + 'antidotos')
        .then(data => data.json())
        .then(data => {
            setAntidotos(data)
        })

    fetch(url + 'toxindromes')
        .then(data => data.json())
        .then(data => {
            setToxindromes(data)
        })
}, [])
```

Figura 13. Programación del fetch de información desde el servidor al cliente.

Se diseñaron tres pestañas para separar las categorías. Para ello se utilizó una fila con una columna que se divide en tres “Tabs” o pestañas.

Se creó un buscador el cual recorre todos los títulos de cada categoría. Para realizar esta función se utiliza un operador lógico de comparación entre la pestaña seleccionada y los caracteres ingresados en la búsqueda. Aquí se integró el reconocimiento de mezcla de mayúsculas o minúsculas en el buscador (Figura 14) como también la comprobación que la cantidad de caracteres recibidos fueran mayor a cero (en el caso contrario se muestran todos los datos).

Bajo cada pestaña se integró un “Accordion”, tipo de pestaña expandible ordenada de manera vertical (ver **¡Error! No se encuentra el origen de la referencia.**). E stos llamados acordeones retornan la información desde los archivos “Antidotos.js”, “Toxicos.js” y “Toxindromes.js”.

Figura 14. Vista del buscador y la sección de acordeones con información de la base de datos. Se puede visualizar el reconocimiento de mayúsculas y minúsculas en el buscador al dar los resultados.

Los archivos nombrados en el párrafo anterior tienen como función asociar los datos obtenidos con las vistas de acordeón creadas. Se integró un botón de “Leer más” para expandir la información usando “Collapse”, una transición para animar esta vista. Se creó una variable booleana de estado para definir cuando está expandido y cuando no (Figura 15).

Se realizó una comparación lógica usando un “and” entre la información obtenida y los campos creados para mostrar la misma, esto quiere decir que, si no se obtienen datos para un campo en específico, este se omite y no se renderiza dentro del “div” del acordeón (Figura 15). El procedimiento anterior se realizó para cada una de las variables de información obtenida en el fetch. Una vez que se realiza la operación se muestran los campos con información dentro del acordeón (ver ejemplo al buscar Carbamazepina en la Figura 16).

```
<Accordion.Item eventKey={index} key={index}>
  <Accordion.Header>{item.name}</Accordion.Header>
  <Accordion.Body className='accordionBody'>
    {
      item.description &&
      <div>
        {item.description}
      </div>
    }
    <div className='text-end'>
      {open && <BiUpArrow onClick={() => setOpen(false)} />}
      {
      !open &&
      <div><BiDownArrow onClick={() => setOpen(true)} /> Leer más</div>
    }
    </div>
```

Figura 15. Programación de comparación de información (verde) y botón de "Leer más" con cambio de estado (rojo).

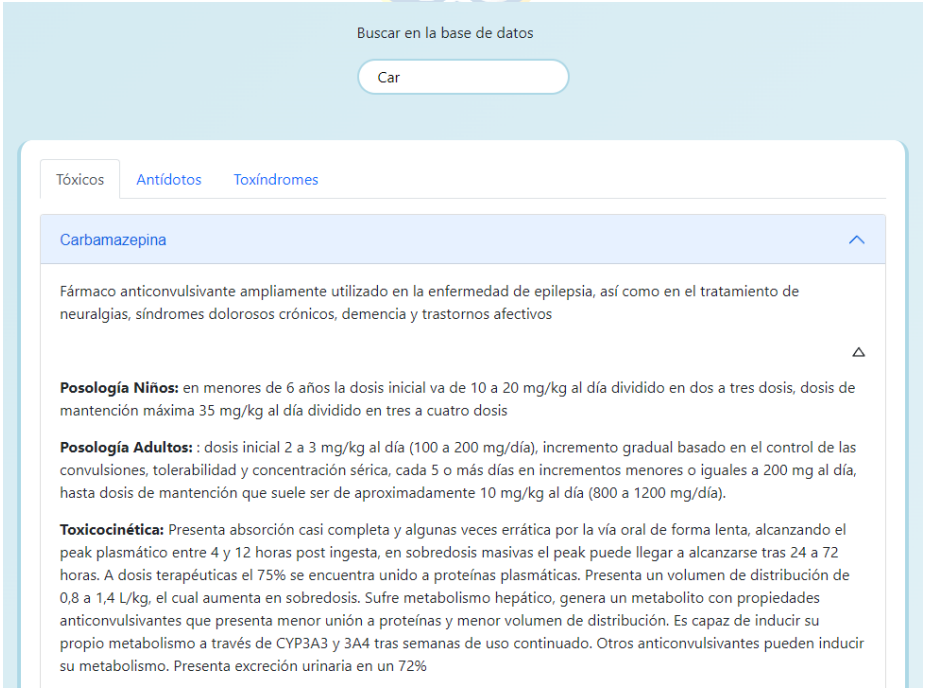

Figura 16. Ejemplo de búsqueda y vista de Carbamazepina en la pestaña de tóxicos.

- Contact: Se utilizó para programar el envío del correo en el formulario de contacto. Mediante la biblioteca Axios se crea una función para enviar la información ingresada por el usuario en el lado del cliente utilizando “axios.post”. Esta función recoge los datos almacenados en la constante “user” y los envía a la dirección del Backend. Se creó un formulario simple utilizando el “input” tag para crear los campos vacíos y se le asignó una constante a cada uno de ellos. El botón de envío se programó para revisar las casillas antes de realizar el post. Para el layout de esta sección se usó una fila con dos columnas y así dar el resultado de la Figura 17 donde se aprecia un recuadro con información de contacto.

Figura 17. Vista de componente de Contacto. Se visualiza el formulario de contacto y un recuadro de información de contacto.

1.7. Programación del Modal de la página de inicio

Se creó una función llamada MyModal (ver Figura 17) con valores booleanos para representar el estado de despliegue de la ventana emergente de atención. Esta se programó para funcionar solo durante la sesión del usuario mediante “sessionStorage”. Si en la sesión actual el estado indica que no se ha mostrado, se niega el estado y con ello se retorna la función del modal. Se configuró el cierre de la ventana mediante un botón de cruz.

```
function MyModal() {
  const [show, setShow] = useState(false);

  const handleClose = () => setShow(false);
  const handleShow = () => setShow(true);
  useEffect(()=>{
    if(sessionStorage.getItem('modal') !== 'shown'){
      handleShow()
      sessionStorage.setItem('modal', 'shown')
    }

  },[])
  return (
    <>
      <Modal
        show={show}
        onHide={handleClose}
        backdrop="static"
        keyboard={false}
        centered
      >
        <Modal.Header closeButton>
          <Modal.Title>¡Atención!</Modal.Title>
        </Modal.Header>
        <Modal.Body>
        La información disponible en esta WebApp está dirigida exclusivamente a profesionales
        y estudiantes de la salud que cuentan con la debida preparación.
        </Modal.Body>
      </Modal>
    </>
  );
```

Figura 17. Programación de ventana emergente de alerta.

Anexo B. Resumen de Memoria de Título

UNIVERSIDAD DE CONCEPCION – FACULTAD DE INGENIERIA
RESUMEN DE MEMORIA DE TITULO

Departamento	**:** Departamento de Ingeniería Eléctrica
Carrera	**:** Ingeniería Civil Biomédica
Nombre del memorista	: Gerardo Alonso Castillo Rodríguez
Título de la memoria	: Desarrollo De Web App Sobre Toxicología Clínica De Apoyo A Personal De Salud
Fecha de la presentación oral	: 26/10/2022
Profesor(es) Guía	: Pamela Guevara, Claudio Müller
Profesor(es) Revisor(es)	: Pablo Aqueveque
Concepto	:
Calificación	:

Resumen

En este documento se entregan los aspectos fundamentales del desarrollo de una aplicación web capaz de ofrecer información valiosa sobre toxicología clínica, ya que, los actuales mecanismos para obtener esta data resultan poco expeditos para el personal de salud y/o público en general.

Se tomaron los datos reunidos en una tesis de Química Farmacéutica de la Universidad de Concepción y se introdujeron a una base de datos. Dentro de los datos reunidos se encuentra el 10% de las intoxicaciones más recurrentes en Chile.

Mediante las tecnologías de MERN, utilizando Visual Studio Code se logró crear una web app. Esta es capaz de mostrar información directamente desde la base de datos subida en la nube.

Se logró la implementación de un recuadro de búsqueda conectado a un servidor para buscar toda la información. Se creó una subpágina con un formulario de contacto para recibir dudas de los usuarios, como solicitar más información referente a los tóxicos.

Se evalúo la funcionalidad de la web app mediante una encuesta hacia profesionales relacionados al área de la toxicología clínica. Los resultados indicaron que la función principal se cumple con éxito y sería una plataforma utilizada por los encuestados.