

**MIGRACIÓN DE LA PLATAFORMA GOPH A REACT JS Y A UNA API
RESTFUL EN NODE.JS PARA SOPORTAR MAYOR NÚMERO DE
CONEXIONES RECURRENTE**

SERGIO ESTEBAN VALENCIA VELÁSQUEZ

1.088.353.404

MSc. ALONSO TORO LAZO

MONITOR

MSc. ALEXANDER DE JESUS CELÍN BARRAZA

TUTOR

UNIVERSIDAD CATÓLICA DE PEREIRA

FACULTAD DE CIENCIAS BÁSICAS E INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS Y TELECOMUNICACIONES

PEREIRA

2021

RESUMEN

En la actualidad, los aplicativos web desarrollados sobre lenguajes solos y con antiguas prácticas están sufriendo por el constante aumento en sus clientes y peticiones, situación que los hace más lentos y requieren un mayor tiempo de respuesta. Inggo Developments S.A.S. ha tenido esta problemática al tener un alza significativa en sus usuarios y peticiones realizadas, trabajando a través de una plataforma que consume recursos de los propios servidores con conectividad muy baja, provocando un colapso en el sistema al tener mayor concurrencia. Para esta problemática se planteó como solución una migración del software a React JS y Node.JS para disminuir el consumo de los propios servidores, aumentando el consumo de los clientes, lo cual permite soportar una mayor concurrencia y estabilidad para el sistema. Este documento presenta de manera breve las actividades realizadas durante dicho proceso de migración.

Palabras clave: Cliente – Servidor, Node.JS, Migración de software, React JS.

ABSTRACT

Currently, web applications developed over lonely languages and with old practices are suffering by the constant increase in their clients and requests, situation that makes them slower and requires a higher response time. Inngo Developments S.A.S. has had this problematic due to a significant rise in its users and requests made in the software, working through a platform which consumes resources of their own servers with a very low connectivity, triggering a collapse on the system because of a higher concurrence. For this problematic it was proposed as a solution a software migration to React JS and Node.JS to reduce the consumption of the clients, which allows to hold a higher concurrence and stability for the system. This document presents briefly the activities made along said migration process.

Keywords: Client – Server, Node.JS, Software migration, React JS.

TABLA DE CONTENIDO

1.	INTRODUCCIÓN	7
2.	DESCRIPCIÓN DEL ESCENARIO DE PRÁCTICA	9
2.1.	Reseña histórica	9
2.2.	Estructura organizacional	9
2.2.1.	Área de intervención	10
3.	DESCRIPCIÓN DE LAS ÁREAS DE INTERVENCIÓN	11
4.	JUSTIFICACIÓN	13
5.	OBJETIVOS	15
5.1.	Objetivo general	15
5.2.	Objetivos específicos	15
6.	MARCO TEÓRICO	16
6.1.	Ingeniería de software	16
6.2.	Metodologías de desarrollo de software	17
6.3.	Modelos de ciclo de vida	19
6.3.1.	Modelo de ciclo de vida en cascada	20
6.4.	Arquitectura de software	21
6.4.1.	Patrón de arquitectura Cliente - Servidor	22
6.5.	Tecnologías de desarrollo de software web	22
7.	DESARROLLO DE LA PRÁCTICA	25
7.1.	Capacitar al equipo de trabajo en la creación de bases de datos en PostgreSQL	26
7.2.	Estudiar el Framework React JS y los lenguajes Javascript, Node.JS y Jsx para la migración del aplicativo GoPH	27

8.	RESULTADOS OBTENIDOS	29
8.1.	Realizar el bosquejo del Frontend en Html y CSS para la migración	29
8.2.	Desarrollar el Backend del aplicativo en Node.JS	36
8.3.	Desarrollar la migración del Frontend a React JS	39
8.4.	Implementar cifrado a cada petición que haga el Frontend al Backend	40
8.5.	Pruebas de software migrado y salida a producción	41
8.6.	Propuesta de un framework genérico para la migración de aplicativos webs	44
9.	CONCLUSIONES	46
10.	RECOMENDACIONES	47
11.	REFERENCIAS	48

LISTA DE ILUSTRACIONES

<i>Ilustración 1. Estructura orgánica INGGO DEVELOPMENTS SAS.</i>	10
<i>Ilustración 2. Algunos modelos y/o metodologías con sus características, ventajas y desventajas.</i>	19
<i>Ilustración 3. Modelo de ciclo de vida en cascada [10].</i>	20
<i>Ilustración 4. Archivos a realizar con la plantilla enviada por el cliente.</i>	26
<i>Ilustración 5. Interfaz de “SQL Manager For PostgreSQL”.</i>	27
<i>Ilustración 6. Login del aplicativo GoPH.</i>	30
<i>Ilustración 7. Home del aplicativo GoPH.</i>	31
<i>Ilustración 8. Menú de votaciones del aplicativo GoPH.</i>	32
<i>Ilustración 9. Informes - Anexos asamblea del aplicativo GoPH.</i>	33
<i>Ilustración 10. Informes - Historia votos del aplicativo GoPH.</i>	34
<i>Ilustración 11. Informes - Mis poderes del aplicativo GoPH.</i>	35
<i>Ilustración 12. Soporte del aplicativo GoPH.</i>	35
<i>Ilustración 13. app.js Backend.</i>	37
<i>Ilustración 14. index.js Backend.</i>	38
<i>Ilustración 15. r.Login.js Backend.</i>	38
<i>Ilustración 16. CopyRight.js Frontend.</i>	40
<i>Ilustración 17. JMeter Hilos para realizar pruebas.</i>	42
<i>Ilustración 18. JMeter Informes Parte 1.</i>	43
<i>Ilustración 19. JMeter Informes Parte 2.</i>	43

1. INTRODUCCIÓN

Inggo Developments S.A.S es una empresa desarrolladora de software que está ubicada en la ciudad de Pereira, Colombia, la cual desarrolló un aplicativo web para Propiedad Horizontal llamado GoPH, el cual se enfoca en ser el medio para la realización de asambleas. Este aplicativo web fue diseñado y desarrollado en la plataforma francesa WinDev, la cual tiene su propio lenguaje de programación y propio framework, siendo de una complejidad baja tanto para desarrolladores como para diseñadores web, sin embargo, es una plataforma con poca comunidad, lo que limita el desarrollo y mejoramiento del aplicativo web.

Por otro lado, al subir el aplicativo web al servidor y ser administrado por el administrador de sitios web ISS, se observó que, con cada conexión de un cliente al aplicativo web, esta consume los recursos del servidor y no del navegador del cliente, el cual sería uno de los grandes problemas que enfrentaría el aplicativo web a la hora de aumentar sus clientes, pues tendría que aumentar su cantidad de servidores o buscar posibles alternativas.

Dada la situación anterior, el ingeniero en jefe toma la decisión de buscar nuevas alternativas, plataformas o framework, que permitan aumentar la concurrencia y disminuir el costo en servidores, infraestructura y redes. Se optó entonces por buscar algunos de los i-frame más usados como Laravel, Angular y React JS. Tras una investigación inicial, se decidió realizar una migración del software de la plataforma WinDev a React JS y una API restful en Node.JS, donde React JS cumplirá el objetivo de consumir los recursos del navegador del cliente sin necesidad de consumir los propios del servidor, y, por otro lado, Node.JS permitiendo tener una gran concurrencia en el servidor actual sin necesidad de aumentar la infraestructura de la organización.

Así, en el presente documento se describe el proceso de migración de la plataforma GoPH del framework WebDev a React JS y Node.JS, donde se implementaron diferentes librerías, métodos y técnicas para lograrlo. Al finalizar el procedimiento de migración, se realizaron las pruebas pertinentes al software de manera que esté listo para producción.

Este texto estará dividido en diferentes secciones. En primer lugar, se encuentra la descripción del escenario de prácticas, sección en la cual se describe el escenario donde se

realizaron las prácticas académicas y en la que se desarrolló el presente proyecto. La segunda sección se refiere a las áreas a ser intervenidas como parte del proceso de prácticas y las actividades realizadas en ellas. La tercera sección es la justificación del problema de la práctica donde se presenta la pertinencia del proyecto. En la cuarta sección se exponen los objetivos del proyecto, mientras que la quinta sección corresponde al marco teórico en el cual se plasman los referentes teóricos que sustentan el trabajo. El desarrollo de la práctica es la sexta sección; allí se muestra el paso a paso del proceso de migración, las pruebas realizadas y se da a conocer la capacitación previa al inicio del proyecto. En la séptima sección se encuentran los resultados obtenidos del trabajo, las conclusiones de este son la octava sección y, para finalizar, se encuentran las recomendaciones del proyecto seguido por las referencias del mismo.

2. DESCRIPCIÓN DEL ESCENARIO DE PRÁCTICA

2.1. Reseña histórica

Inggo Developments SAS en adelante (Inggo Dev) es una empresa fundada en el año 2014 y registrada formalmente en noviembre de 2019 por Elkin Huber Álvarez Vásquez y Alexander de Jesús Celín Barraza, buscando crear una empresa regional de desarrollo de software.

El principal objetivo de la empresa es desarrollar software y distribuirlo entre las empresas o clientes en diferentes ciudades del país, y a nivel internacional en países como Brasil, México y España. Actualmente la empresa tiene tres aplicativos de software registrados, Spectrum, Parky y GoPH, el último de ellos ha venido evolucionando a pasos agigantados y es un software que consiste principalmente en facilitar la realización de asambleas virtuales, ya que con la situación actual por la pandemia Covid-19 se han migrado las asambleas presenciales a virtuales, razón por la cual el software se ha ido adaptando a la necesidad de los clientes y requerimientos establecidos por la ley.

2.2. Estructura organizacional

Con respecto a los procesos de la empresa, éstos se pueden dividir principalmente en tres secciones, la parte comercial, la administrativa y la de desarrollo. Por lo general, cada cliente primero llega por la parte comercial de la empresa realizando una compra, acto seguido se solicita información requerida para crear la base de datos del cliente, la cual es realizada por el equipo de desarrollo e implementada en los servidores de la compañía, con su respectivo proyecto para que el cliente pueda trabajar. Por otra parte, la sección administrativa se encarga de facturar y gestionar la coordinación entre el cliente y la empresa Inggo Dev. Finalmente, la sección de desarrollo se encarga de realizar la base de datos del cliente, generar el proyecto, crear una cadena de conexión para el cliente y mantener actualizando el software; por otra parte, esta sección también se encarga de los procesos de migración de los aplicativos.

Los cargos en la empresa están distribuidos de la siguiente manera, ver **Ilustración 1**:



Ilustración 1. Estructura orgánica INGGO DEVELOPMENTS SAS.

2.2.1. Área de intervención

El ejercicio académico correspondiente a la práctica profesional se desempeña en el cargo de Ingeniero de desarrollo, a cargo del proceso de migración del aplicativo desarrollado en WebDev hacia React JS con un backend en Node.JS; aprendiendo el lenguaje Javascript y JSX, desarrollando la lógica tanto del backend como del frontend, conectando el backend con la base de datos en PostgreSQL y haciendo peticiones get de tipo ajax desde el frontend hacia el backend. También correspondiente a la práctica, se tiene el rol de coordinador de la realización de las bases de datos, y dando soporte a la plataforma en caso de tener inconvenientes o actualizaciones requeridas durante el periodo de prácticas académicas.

3. DESCRIPCIÓN DE LAS ÁREAS DE INTERVENCIÓN

El escenario de práctica ha tenido a lo largo de su desarrollo una problemática muy común en los sistemas hoy en día, el límite de conexiones simultáneas que permite tener una tecnología de desarrollo web. En un inicio, esto puede parecer poco importante, sin embargo, con el pasar del tiempo y el crecimiento del software se logra tener miles de personas conectadas, lo cual reduce considerablemente la velocidad del sitio web o, en el peor de los casos, genera el colapso de la página web.

Por este motivo, el área de Desarrollo de Software ha visto la necesidad de investigar y buscar nuevas tecnologías que sean capaces de asimilar mejor la carga de conexiones en el mismo instante, para migrar el aplicativo GoPH a estas nuevas tecnologías y que su funcionamiento sea, en lo posible, el más óptimo, con los recursos que se tienen, a la hora de tener más conexiones simultáneamente.

Por otro lado, al aumentar la demanda del software, aumentan los clientes y consumidores finales del software, por lo cual el equipo de trabajo se aumenta. Esta situación ha causado que a lo largo de la realización de bases de datos de los clientes haya descoordinación, por lo cual el ingeniero en jefe, dentro del escenario de prácticas, no puede coordinar y/o supervisar la realización de dichas bases de datos y cumplir a cabalidad con las tareas asignadas a la hora de crear el sistema para cada cliente.

El proceso de realizar las bases de datos comienza desde el recibimiento de una plantilla en formato excel, la cual cada cliente debe enviar con la información de los usuarios para poder proceder a crear el sistema; usualmente los clientes no envían la información dentro de esta plantilla, por lo que es necesario devolverla e iniciar nuevamente el proceso; al llegar la plantilla correctamente, es necesario pasar esta información a dos archivos CSV, donde se crean las tablas de usuarios e inmuebles, para posteriormente ser subidas a la base de datos por medio del programa “SQL Manager For PostgreSQL”, donde, cabe aclarar, la base de datos está hecha en PostgreSQL.

Para todo este proceso no hay una persona a cargo, generando descoordinación, inconformidad por parte del cliente y falta de control sobre cada cliente nuevo que se crea o del que llega al sistema para ser actualizado.

4. JUSTIFICACIÓN

En la actualidad, el desarrollo de software busca tener la facilidad de tener un entorno de desarrollo de software de menos complejidad y mayor agilidad, donde solo sea necesario conocer un lenguaje para desarrollo, diseño y conexión a bases de datos, para no verse en la necesidad de tener los diferentes procesos en lenguajes o plataformas distintas. Es por esto que grandes empresas como Facebook o Google, han buscado desarrollar tecnologías, herramientas, librerías, etc. propias, para lograr este objetivo y tener plataformas web más estables, con mayor capacidad y mejor optimizadas.

Facebook, por ejemplo, creó un framework llamado React JS, el cual se puede considerar una librería para Node.JS, que permite combinar el entorno de HTML, CSS y JS, en un lenguaje llamado JSX. Este framework facilita la realización de aplicaciones web, agradables y de fácil manejo para el usuario, haciendo de esta una herramienta adecuada para el fácil y rápido desarrollo de aplicaciones web.

Por su parte, Inggo Developments S.A.S desarrolló su aplicativo GoPH en la plataforma francesa WinDev, la cual, si bien tenía una dificultad baja para desarrolladores y diseñadores web con su propio lenguaje de programación, tiene poca comunidad que limita su desarrollo y mejoramiento del aplicativo. Asimismo, el problema más relevante de esta plataforma es el consumo de recursos del servidor y no del cliente final, lo que conlleva a buscar alternativas que permitan mejorar el rendimiento del aplicativo y no consumir sus propios recursos.

Así, se presenta de forma clara la necesidad de realizar el proceso de migración del aplicativo web de WinDev a React JS y Node.JS para dar cumplimiento al objetivo de la empresa de aumentar la concurrencia y disminuir el costo en servidores, infraestructura y redes al consumir los recursos del navegador del cliente sin necesidad de consumir los propios del servidor. De esta forma, es pertinente la consignación del proceso de migración por medio del presente trabajo, que permita detallar el desarrollo de dicho procedimiento.

Cabe aclarar que la mitigación busca reducir el estrés de los servidores para no aumentar la planta física de la empresa, sino que, por medio del desarrollo, se logre un mejor rendimiento

del sistema. Esto, mediante el aprovechamiento de tecnologías que generan notablemente mayor estabilidad y seguridad en los desarrollos de software.

Adicionalmente, este proyecto permitirá a la empresa brindar mejor calidad en el software, lo cual se traduce en una mejor oferta a los clientes, con estabilidad constante, menor propensión a fallos y mayor capacidad de conexiones.

5. OBJETIVOS

5.1. Objetivo general

Migrar la plataforma GoPH de WebDev a React JS y a una Api Restful en Node.JS para soportar mayor número de conexiones simultáneas.

5.2. Objetivos específicos

- Desarrollar un backend en Node.JS conectándose a una base de datos existente en Postgresql.
- Implementar peticiones Get desde React JS con Axios hacia el backend.
- Fusionar plantilla en Html5 en React JS con JSX y su respectiva lógica.
- Realizar las pruebas necesarias para verificar el correcto funcionamiento de la aplicación una vez finalizada la migración.

6. MARCO TEÓRICO

En esta sección se presentan los principales conceptos teóricos en los que se enmarca el desarrollo de este proyecto, de forma que las actividades prácticas realizadas durante el proceso de migración del aplicativo WINDEV a REAC JS esté sustentado en buenas prácticas que le confieran características de calidad. De esta forma, se contemplan esencialmente los conceptos de ingeniería de software, metodologías del desarrollo de software, modelos de ciclo de vida (en particular el modelo de ciclo de vida en cascada utilizado en el presente proyecto), arquitectura de software, patrón de arquitectura cliente - servidor y, por último, las tecnologías de desarrollo web implementadas.

6.1. Ingeniería de software

Según Pressman, en el libro “Ingeniería de software: Un enfoque práctico” [1], la ingeniería de software incluye los procesos, métodos y herramientas para administrar y hacer ingeniería con el software. Por otro lado, IEEE en [2] define la ingeniería de software como la aplicación de un enfoque sistemático, disciplinado y cuantificable aplicado en el desarrollo, operación y mantenimiento de software; es decir, la aplicación de la ingeniería al software. Como lo explican Toro & Cardona en [3], la ingeniería de software se puede entender como “una disciplina o área de la Informática o Ciencias de la Computación, que ofrece métodos y técnicas para desarrollar y mantener software de calidad que resuelven problemas de todo tipo”.

De acuerdo con las definiciones anteriores, es posible asumir que la ingeniería de software es un conjunto de herramientas, procesos, técnicas y métodos que son aplicados para el desarrollo, administración y mantenimiento de un software durante toda su vida, siendo esto la aplicación de la ingeniería al software para brindar un software de alta calidad.

Por lo general, para aplicar ingeniería en el software se utiliza una metodología de desarrollo de software específica que permita la organización de las actividades a realizar de manera ordenada, de tal manera que se logre un desarrollo de software con alta probabilidad de éxito. Por tal motivo, se presenta en la siguiente sección una indagación sobre las metodologías de desarrollo de software.

6.2. Metodologías de desarrollo de software

En [4], Espinoza define Metodología como un marco de trabajo que comprende las actividades, procedimientos, técnicas, herramientas y documentación, añadiendo que estas metodologías indican un plan adecuado de gestión y control del proyecto que se llevará a cabo.

Las metodologías para desarrollo de software son, según Ruiz et al. [5], “el modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito”, cabe aclarar que esto comprende desde la etapa de idear, implementar y mantener el software desde que este surge hasta que cumple con el objetivo por el cual fue creado.

Lo anterior, permitiría definir un plan adecuado para la gestión y control del proyecto, indicando el qué, cuándo y cómo se debe hacer, dando los lineamientos necesarios para llevar el desarrollo de un proyecto para tener altas posibilidades de realizarse satisfactoriamente [1, 3].

En la actualidad existen diversas Metodologías como las Metodologías de Análisis y Diseño Estructurado, las Metodologías Orientadas a Objeto, las Metodologías de Desarrollo Ágil y las Metodologías Tradicionales, cada una es diferente y se utilizan según el proyecto a realizar y la necesidad del cliente. La Metodología de Análisis y diseño estructurado consiste en la división de las funciones y datos, lo cual crea dos ramas, una orientada a los procesos y otra a los datos.

Por otra parte, la Metodología Orientada a Objetos consiste en priorizar la identificación de cada objeto que tendrá el software para luego especificar el comportamiento que tendrá cada uno.

También se encuentran las metodologías ágiles que valoran al individuo y las iteraciones del equipo más que a los propios procesos o herramientas a utilizar, en ellas se considera más importante el desarrollo de software y la respuesta rápida a un problema que la misma documentación del proyecto, dando esto más importancia a la documentación como el código fuente [4].

Por último, las Metodologías Tradicionales se centran principalmente en llevar una documentación exhaustiva de todo el proyecto, planificación y control del mismo, en las especificaciones exactas de los requisitos y modelado y en cumplir un plan de trabajo desde la fase inicial del desarrollo del proyecto [6].

En la siguiente figura, **Ilustración 2**, se pueden evidenciar algunas de las metodologías actualmente utilizadas con sus características, ventajas y desventajas.

ENFOQUE	NOMBRE	PRINCIPALES CARACTERÍSTICAS	VENTAJAS	DESVENTAJAS
Tradicional	Prototipado	<ul style="list-style-type: none"> - Es una metodología de ensayo y error. - Si un requerimiento levantado no es lo esperado, se vuelve a iniciar. - Es una metodología evolutiva. 	<ul style="list-style-type: none"> - No se modifica el flujo del ciclo de vida. - Se construye un software que satisfaga las necesidades del cliente. - Alta probabilidad de éxito del software. 	<ul style="list-style-type: none"> - Alta probabilidad de aumentar el tiempo de entrega. - Se pueden aumentar los costos del proyecto. - No se entrega hasta culminar con cada requerimiento del cliente.
Tradicional	Espiral	<ul style="list-style-type: none"> - Se hacen entregas del proyecto progresivamente. - Se pueden levantar requerimientos en el camino. - El crecimiento del software es progresivo. 	<ul style="list-style-type: none"> - El riesgo del software es poco. - El desarrollo es interactivo. - Es fácil incorporar nuevas funcionalidades. 	<ul style="list-style-type: none"> - No hay una duración de ejecución establecida. - Un mal análisis de riesgos puede influir negativamente a todo el proyecto.
Tradicional	Incremental	<ul style="list-style-type: none"> - Es una combinación de la metodología en cascada y la interactiva. - Cada incremento se realiza al terminar un requerimiento. - Se entrega al cliente progresivamente. 	<ul style="list-style-type: none"> - Facilidad para probar y depurar. - Cada interacción es un hito gestionado fácilmente. - Facilidad para gestionar los riesgos. 	<ul style="list-style-type: none"> - Cada fase de una interacción es rígida y no se superpone con otras. - Pueden surgir problemas al surgir nuevos requisitos porque estos son definidos al inicio.
Ágil	Kanba	<ul style="list-style-type: none"> - Fácil implementación. - Se ayuda mucho de ayudas visuales para las tareas. - Se realiza una tarea hasta acabarla. 	<ul style="list-style-type: none"> - Se acortan los tiempos de entrega. - No se requiere una gran planificación para implementarlo. - Se puede llevar un control más fácil del código y los 	<ul style="list-style-type: none"> - No se implementa bien en ciclos de vida muy largos. - No se adapta fácilmente a cambios en la marcha.
Ágil	Scrum	<ul style="list-style-type: none"> - Se entrega progresivamente el producto. - Es interactivo e incremental por cada uno de sus sprints. - Se caracteriza por su rápida respuesta a soluciones o necesidades. 	<ul style="list-style-type: none"> - Flexibilidad y adaptación a los contextos. - Gestión sistemática de riesgos. - Resultados anticipados. 	<ul style="list-style-type: none"> - Funciona en equipos reducidos. - Requiere una exhaustiva definición de las tareas y sus plazos.
Ágil	Programación extrema (XP)	<ul style="list-style-type: none"> - Comunicación con el cliente. - Planificación flexible y abierta. - Rápida respuesta a cambios. 	<ul style="list-style-type: none"> - Software estable debido a sus continuas pruebas. - Aplicación rápida a los cambios. - Menos errores gracias a la programación en pareja. 	<ul style="list-style-type: none"> - Se requiere mucho tiempo. - Requiere control de versiones. - Altos costos para el proyecto.

Ilustración 2. Algunos modelos y/o metodologías con sus características, ventajas y desventajas.

Fuente: Elaboración propia.

La metodología de desarrollo de software utilizada en un proyecto puede ser implementada de acuerdo con un modelo de ciclo de vida del software, concepto que será tratado a continuación.

6.3. Modelos de ciclo de vida

En primer lugar, se debe aclarar que el término “proceso” se refiere a un conjunto ordenado de actividades y una serie de pasos que involucran tareas, restricciones y recursos que producen algo, por tanto, es un marco de trabajo de estas tareas. Así, el modelo de ciclo de vida se puede definir como el proceso de ciclo de vida del software.

Por otro lado, un ciclo de vida es aquel que comprende toda la vida del software desde su captura o identificación de los requisitos del proyecto hasta el mantenimiento y futuras mejoras. Por lo tanto, un modelo de ciclo de vida son los procesos involucrados en el desarrollo del software [7], por lo cual se puede decir que el modelo de ciclo de vida es el marco de trabajo que tendrá el análisis, diseño, fabricación, prueba, implementación y mantenimiento del software, y cada una de las etapas internas que tengan estos.

El modelo de ciclo de vida es el marco de referencia para un proyecto que contiene los procesos, actividades y tareas involucradas en el desarrollo [7], esto abarca desde el inicio del proceso con la adquisición de los requisitos, luego el marco de trabajo de cómo se realizará con las tareas que comprenda y, por último, con la finalización del desarrollo al ponerlo a la disposición. Esto es la explotación y el mantenimiento del producto realizado, donde se encuentra el versionamiento y mantenimiento del software durante su vida útil [8]. En el presente trabajo se adoptó la utilización del modelo de ciclo de vida en cascada con el cual se trabajó durante la práctica académica, el cual se explica a continuación.

6.3.1. Modelo de ciclo de vida en cascada

El modelo de ciclo de vida en cascada es un procedimiento secuencial que permite avanzar entre etapas hasta finalizar la anterior [7, 9], dichas etapas se pueden evidenciar en la **Ilustración 3.**

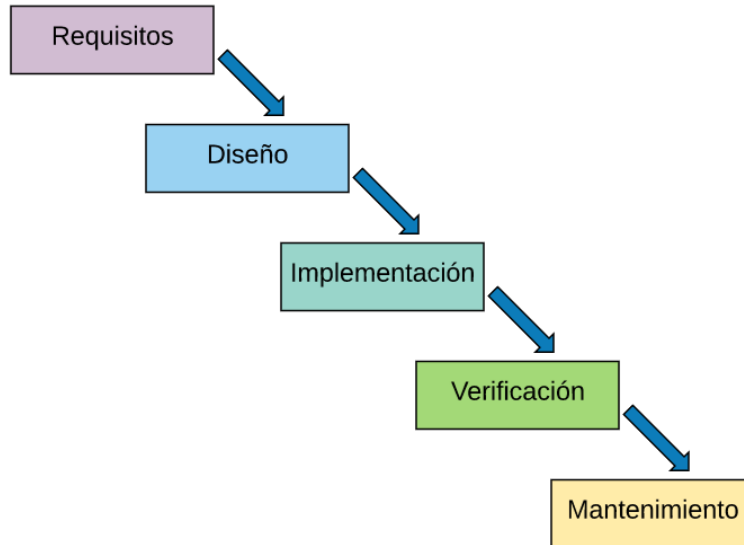


Ilustración 3. Modelo de ciclo de vida en cascada [10].

Los requisitos del proyecto son aquellos que el cliente indica a la hora de la entrevista, los cuales se capturan o identifican realizar el análisis y organizarlos, y determinar su viabilidad para, de esta forma, dirigirlos a la siguiente etapa del ciclo de vida, o en su caso contrario ser descartados antes de ser pasados para la siguiente etapa; la siguiente etapa sería el diseño de cada uno de estos. Como se menciona, primero se analiza cada uno de los requisitos anteriormente identificados o capturados, así realizar el análisis de su viabilidad, complejidad y tecnologías o técnicas de programación a utilizar, y de esta manera diseñar de la mejor manera la codificación y/o implementación de la arquitectura o tecnologías a utilizar en el requisito para el proyecto.

Por otro lado, cada requerimiento ya analizado y diseñado que es viable para el proyecto es enviado a la siguiente etapa del ciclo de vida, la implementación, donde es pasado el requisito con su respectivo análisis y diseño a los codificadores para realizar la codificación necesaria para implementar el diseño y funcionamiento del requerimiento. Todos los diseños

entregados son codificados para pasar a la verificación, la cual está comprendida por las pruebas que se realizan a los requisitos ya codificados para determinar si están listos para salir a producción o si hay errores por corregir o mejorar y, de esta manera, lanzar un software robusto y con la cantidad mínima de falencias posibles.

Por último, se encuentra el mantenimiento del software, el cual se empieza a realizar una vez lanzado el software a producción, este abarcaría posibles actualizaciones, correcciones o mejoras, todo esto durante la vida útil del software.

El problema más grande que este modelo tiene es el tiempo, ya que no se puede hacer un entregable hasta finalizar todas las etapas y para un proyecto de gran magnitud significa más costo [10].

Al igual que sucede con la metodología de desarrollo y el modelo de ciclo de vida, la elección de una arquitectura de software adecuada para las características del proyecto es un fundamental para lograr productos de calidad, ya que constituye la estructura sobre la cual se realizará el despliegue del software desarrollado. Por ello, este concepto es presentado en la siguiente sección.

6.4. Arquitectura de software

En términos generales, se entiende a la Arquitectura como “la proyección, diseño y construcción de espacios habitables por el ser humano” [11]; es por esto que la ingeniería de software implementó su propia arquitectura para desarrollar software, ya que no existía un orden que seguir o una organización en los proyectos. La IEEE, en su estándar 1471 [12] define arquitectura de software como la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.

Esta definición da paso a comprender el tamaño e importancia que tiene una arquitectura de software en la organización de este para que sus componentes no se mezclen, sino que tengan en cuenta una arquitectura por la cual el programa se guíe y funcione. Tener en cuenta esto no solo facilita el entendimiento del software, sino que facilita sus procesos, permitiéndole mejorar el rendimiento y disminuir el tiempo de ejecución y respuesta de sus acciones.

De acuerdo a [13] “Una arquitectura de software para un sistema es una estructura o estructuras del sistema, que consiste en elementos, sus propiedades externamente visibles y sus relaciones entre ellos”, así, esta propone patrones de arquitectura de diferentes características que consisten principalmente en proporcionar conjuntos de subsistemas que especifican las funciones a cumplir de cada uno con sus respectivas normativas o reglas para tener una mayor estructuración del software y robustez del mismo. Algunos de los más utilizados son control centralizado, filtro de tubería, cliente-servidor, pizarra, etc; para este trabajo se implementa uno de estos patrones de arquitectura, Cliente Servidor, el cual es explicado a continuación.

6.4.1. Patrón de arquitectura Cliente - Servidor

La arquitectura Cliente-Servidor (C/S) es una arquitectura de desarrollo de software donde las peticiones se reparten entre los servidores que son los encargados de recibir la información, procesarla y guardarla, a lo cual se le llama backend. Por otro lado, su contraparte es el frontend, el cual es la interfaz que está interactuando con los consumidores de la información que realizan las peticiones, los cuales serían el cliente [14].

Esta arquitectura de software permite conectar una variedad de equipos para trabajar coordinadamente con el fin de lograr las peticiones del cliente; en otras palabras, permite tener una red de equipos en el frontend conectados a un backend, el cual está conformado por otros equipos, para que desde el frontend se lancen las peticiones al backend y este, en su red de equipos, pueda leer, procesar y dar una respuesta a dicha petición sin necesidad de que el frontend tenga el más mínimo contacto con el backend, proporcionando una mayor seguridad y confiabilidad del sistema.

6.5. Tecnologías de desarrollo de software web

En la actualidad los desarrollos de software para equipos de cómputo están siendo cada vez menos demandados porque las empresas y clientes desean tener acceso a los recursos del software desde cualquier lugar y en cualquier hora, sin necesidad de depender siempre del mismo dispositivo, es por esto que hoy en día los sistemas de software más demandados son

los aplicativos webs. Estos son un desarrollo de software que se aloja en un servidor y se permite acceder a él por medio del navegador utilizando internet, esto ha hecho que las páginas web pasaran de ser solo unas páginas informativas, a ser espacios interactivos con el cliente como un aplicativo desde internet.

Para desarrollar estos aplicativos webs y sacarlos a producción hay diversas tecnologías, herramientas, lenguajes de programación, iframes, frameworks y servicios en línea. Las tecnologías de desarrollo web y herramientas utilizadas durante la realización del presente trabajo se describen de manera muy breve a continuación:

- NPM (Node Package Manager): Es un sistema de gestión de paquetes y/o librerías de Javascript, mayormente utilizado para la gestión de Node.JS. Este consiste en tres distintos componentes los cuales son el sitio web, una interfaz de línea de comandos y un repositorio de línea que alberga paquetes de Javascript [15].
- Node JS: es un entorno de tiempo de ejecución de JavaScript, el cual es instalado por medio de NPM. Este software es de desarrollo libre y es el que permite ejecutar una aplicación escrita en JavaScript. El cual facilita el desarrollo de una Api Restful del lado del servidor [16].
- React JS: es un Framework de software libre que es mantenido por Facebook y por la comunidad de desarrolladores libres. Este framework es una librería hecha en Javascript y es gestionada por NPM. Se utiliza para diseñar interfaces de usuario y tiene como objetivo principal facilitar el desarrollo de aplicativos webs en una sola página [17].
- HTML: es un lenguaje de etiquetas para realizar el bosquejo de una página, el cual permite realizar la maquetación de una página web [18].
- CSS: es un lenguaje de estilos que se encarga de mejorar el aspecto gráfico a una página web por medio de clases, etiquetas o identificadores [19].
- Bootstrap: es una librería de código abierto, la cual facilita el trabajo a la hora de realizar un diseño de una página web porque son librerías de estilos ya hechas para mejorar el aspecto gráfico en un menor tiempo [20].

- PostgreSQL: es un sistema de gestión de bases de datos gratuito, está basado en lenguaje SQL relacional orientado a objetos para hacer bases de datos [21].
- SQL Manager For PostgreSQL: es un programa que facilita la gestión de las bases de datos en PostgreSQL mediante una interfaz fácil para utilizar y gestionar los datos, consultas, procedimientos, entre otros [22].
- JMeter: es un software de código abierto que se utiliza para realizar carga sobre el comportamiento funcional de prueba y medir el rendimiento de una página web [23].
- NGINX: es un servidor web, proxy inverso, servidor de proxy para correos electrónicos e incluso sirve como firewall, es una plataforma multifuncional y poderosa, de software libre y código abierto [24].

7. DESARROLLO DE LA PRÁCTICA

En la migración de software realizada se utilizaron las tecnologías React JS y Node.JS, implementando un modelo de ciclo de vida en cascada. Esto porque al realizar la migración fue necesario terminar cada una de las etapas para poder continuar con la siguiente, de esta forma tener un software más estructurado a la hora de sacarlo a producción.

En el análisis realizado por la compañía se encontró pertinente migrar el Software a React JS y Node.JS, y utilizar la arquitectura cliente servidor para mayor seguridad y un software más robusto.

Las herramientas implementadas, durante el diseño, para la nueva plataforma fueron HTML, CSS y Bootstrap, elegidas para desarrollar una vista más agradable al usuario y fácil de manejar.

Al iniciar el desarrollo del backend se utiliza la tecnología Node.JS, con las librerías express, pg, pg-hstore, sequelize, morgan, @babel/polyfill, @babel/core, @babel/cli y @babel/preset-env, las cuales permitieron realizar la Api Restful.

Para desarrollar el frontend se utilizó el framework React JS, con las siguientes librerías, npm@latest, create-react-app, react-router-dom, moment react-moment, axios, bootstrap, react-bootstrap, jquery popper.js, sweetalert, save simple-react-validator y react-detect-offline.

Las actividades realizadas durante la práctica académica, asignadas por el ingeniero en jefe, se presentan a continuación:

- Capacitar al equipo de trabajo en la creación de bases de datos en PostgreSQL.
- Estudiar el Framework React y los lenguajes JavaScript, Node.JS y JSX para la migración del aplicativo GoPH.
- Realizar el esqueleto del Frontend Html y CSS, para la migración.
- Desarrollar el backend del aplicativo en Node.JS.
- Desarrollar la migración del frontend a React.
- Implementar cifrado a cada petición que haga el frontend al backend.
- Lanzar a producción el resultado del software migrado.

Cada una de estas actividades tiene un conjunto de sub-actividades, las cuales también fueron realizadas en el ejercicio académico y se presentarán de la siguiente manera: las actividades formación realizadas al inicio de la práctica se presentan en el apartado siguiente; las actividades relacionadas específicamente con el proceso de migración del software, que hacen parte del ciclo de vida del desarrollo, se presentarán en el capítulo 8 (resultados obtenidos).

7.1. Capacitar al equipo de trabajo en la creación de bases de datos en PostgreSQL

Debido al aumento de la demanda en asambleas virtuales a causa de la emergencia sanitaria por el Covid-19, se ha tenido que aumentar el personal para la realización de bases de datos y asignar un coordinador encargado de este equipo para que no haya inconvenientes, inconformidades de los clientes y mucho menos pérdida de información.

En el proceso, se les explica a dos nuevos integrantes del equipo de sistemas cómo trasladar los datos de la plantilla del cliente a los archivos CSV correspondientes, para posteriormente importarlas a sus respectivas tablas en la base de datos. Estos archivos se pueden observar en la **Ilustración 4**. Con esta información ya se pueden crear los archivos CSV necesarios para la creación de las bases de datos.

ITEM	TIPO	NOMBRES	LOGIN	CLAVE	EMAIL	TELEFONO	INMUEBLE	ESTADO	FECHA
1	-99	2	ADMINISTRA	Goph	XcPM7Appb+na	0	-99	1	6/4/2021
3	1	2			wmrPGwXdufna	0	1	1	6/4/2021

Inmuebleid	direccion	coeficiente
1	-99	ADMIN
2		1
3		2
4		3

Ilustración 4. Archivos a realizar con la plantilla enviada por el cliente.

Al finalizar cada archivo se continúa con el siguiente paso del proceso, que corresponde a la creación y subida de la base de datos por cliente. Aquí es donde se utiliza el programa “SQL Manager For PostgreSQL” (Ver **Ilustración 5**), el cual facilita el proceso de crear la base de

datos porque todo lo hace visible a través de una interfaz amigable con el usuario, no por medio de código SQL.

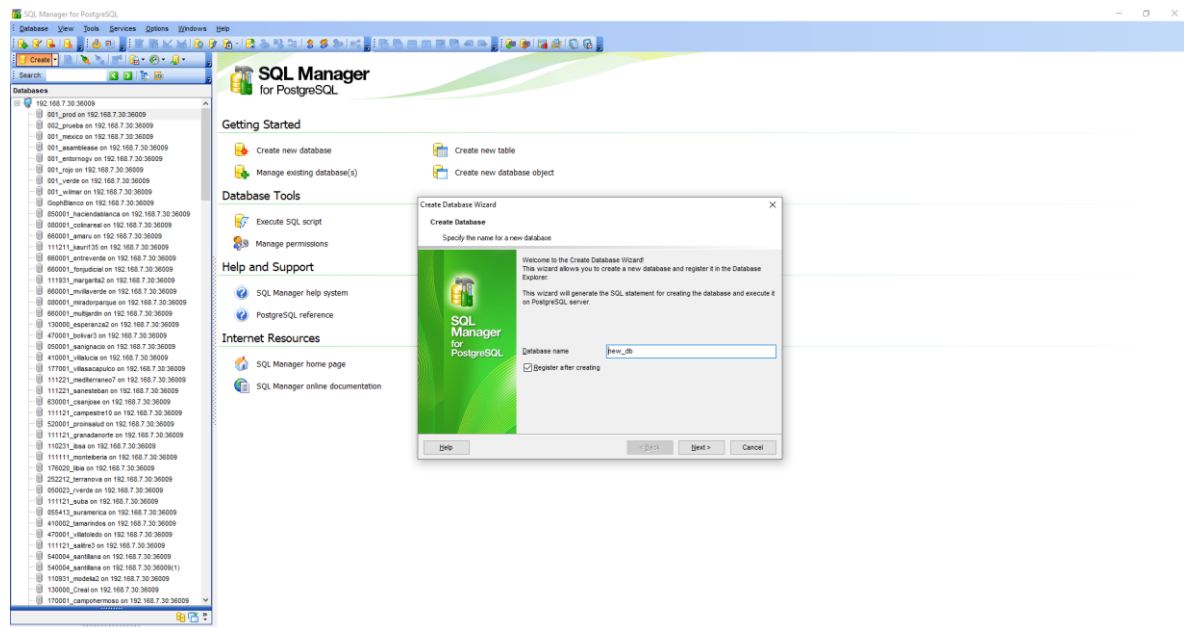


Ilustración 5. Interfaz de “SQL Manager For PostgreSQL”.

La importación de cada archivo CSV en su respectiva base de datos, es también facilitado por el programa “SQL Manager For PostgreSQL”, ya que a través de la misma interfaz gráfica se puede importar cada uno de los archivos. Esto se realiza para ambos archivos CSV, para la importación de las tablas de suscriptor e inmueble, porque todo conjunto, que corresponde a los clientes, es diferente.

Se crea la cadena de conexiones a través de un archivo CFG, en el cual se encuentran los datos de conexión a la base de datos.

7.2. Estudiar el Framework React JS y los lenguajes Javascript, Node.JS y Jsx para la migración del aplicativo GoPH

Tras la decisión tomada por el ingeniero en jefe del departamento de sistemas se inicia con el estudio de estas nuevas tecnologías con un curso en línea de Udemy, hecho por Victor Robles [1], en el cual enseña Node.JS, MongoDB, Angular, React y VueJS. Se realizó el curso completo de Node.JS y React JS, ya que estas fueron las tecnologías estudiadas por el ingeniero en jefe, como solución al problema de conexiones recurrentes en GoPH.

Se realiza todo el curso y los ejercicios propuestos en el mismo, para aprender y afianzar los conocimientos impartidos sobre las tecnologías, React JS y Node.JS, también se empiezan a crear pequeñas partes del código principal con los ejemplos mostrados en las clases para darse una idea de la magnitud del escenario y su complejidad.

Este proceso inició el 1 de enero del presente año y finalizó el 30 de enero del mismo año; con la finalización satisfactoria del curso se dio inicio a la siguiente etapa de las tareas asignadas en el escenario de prácticas académicas.

8. RESULTADOS OBTENIDOS

Son presentadas en esta sección los resultados de la ejecución de las actividades desarrolladas como parte del proceso de migración del software, las cuales están relacionadas con el diseño, codificación y pruebas del aplicativo resultante, iniciando por el frontend y finalizando con el backend.

8.1. Realizar el bosquejo del Frontend en Html y CSS para la migración

Al decidir cambiar de FrameWork el equipo de trabajo optó por cambiar el estilo de la vista y reformar un poco la interfaz con el usuario para que pareciera aún más amigable con el usuario final, esto permitiendo que el usuario tenga una mejor experiencia con el programa y sea de mayor facilidad su manejo, considerando que el mayor porcentaje de público que se atiende mediante la aplicación es mayor de 40años.

El bosquejo principalmente está dirigido por el Ingeniero en jefe, y jefe inmediato de las prácticas, el cual realiza algunos diseños gráficos para discutir y decidir cómo será la mejor vista para el usuario, con la mayor facilidad de uso posible y con características responsive para optimizar el código. Para el Login se decide por un bosquejo sencillo que solicite la información necesaria para el ingreso a la plataforma (Ver **Ilustración 6**), y un popup con un número de soporte técnico para aquellas personas que lo requieran.

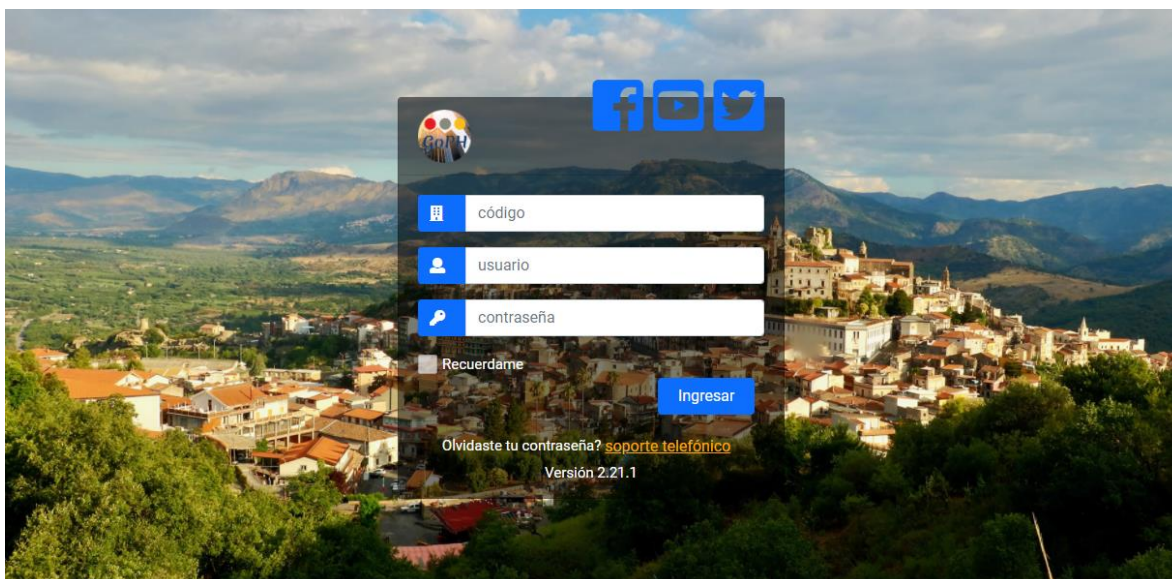


Ilustración 6. Login del aplicativo GoPH.

En todo el diseño del Frontend son utilizadas las librerías de Bootstrap para facilitar el diseño y hacerlo ver más agradable al usuario.

Al ingresar a la plataforma, se diseñó un popup de bienvenida para indicarle al usuario cómo debe registrarse para la asamblea virtual, al encontrarse registrado le aparecerá un nuevo popup que lo dirigirá a la sala de Zoom, mediante la cual podrá asistir a la video Asamblea; en el home de GoPH se pueden apreciar algunos datos del usuario tales como el nombre, conjunto y dirección del inmueble, también el estado de su asistencia en la asamblea, una etiqueta link para dirigirse a la sala de Zoom de ser necesario nuevamente, y un botón mediante el cual podrá acceder al menú para votar.

En la parte superior del aplicativo se encuentra el menú, el cual se podrá observar en cada una de las ventanas a las que acceda el usuario, este menú se realizó responsive para al acceder desde un dispositivo móvil aparezca un menú hamburguesa, para hacerlo más visible se colocó la palabra “Menú”. Para finalizar en la parte inferior del Home se encuentra el footer con toda la información de la empresa Inngo Developments (Ver **Ilustración 7**).

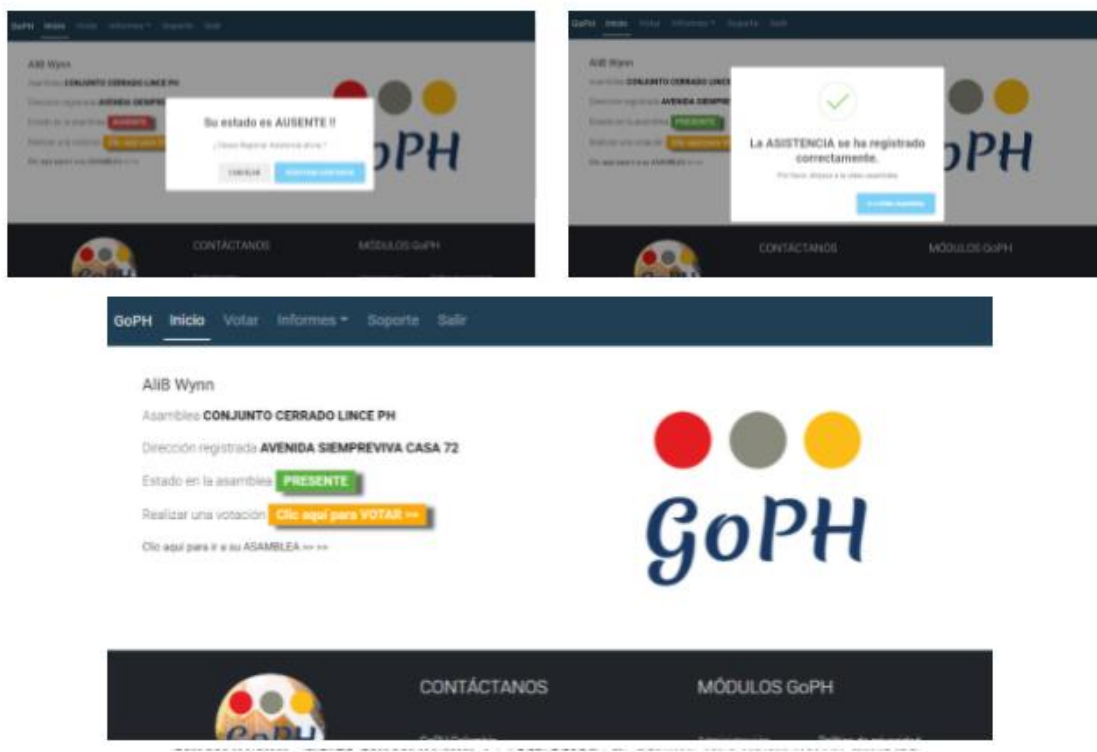


Ilustración 7. Home del aplicativo GoPH.

En la parte superior del menú de votaciones se encontrará la pregunta, en el medio de la ventana las opciones correspondientes a la pregunta, donde al seleccionar una de estas opciones se diseña para que sea resaltada con un check de color verde, de esta forma el usuario puede saber cuál opción tiene seleccionada, y al final de la ventana se encuentran dos botones, uno para votar y otro para cancelar, así a la hora de realizar la votación saldrá un popup de confirmación por la opción a la cual desea votar, el cual tiene los botones de aceptar o cancelar (Ver **Ilustración 8**).

The image shows two overlapping screenshots of the GoPH application's voting interface. The top screenshot displays the 'Registrar votación' screen with a teal header bar containing navigation links: 'GoPH', 'Inicio', 'Votar', 'Informes +', 'Soporte', and 'Salir'. Below the header, a teal box contains the text 'Texto de la pregunta'. A white box below that contains the text 'Apruebo el orden del día anteriormente presentado'. A grey box below that contains the text 'Seleccione la opción de su preferencia.' Below these boxes are three input fields with labels 'SI', 'NO', and 'VOTO EN BLANCO'. At the bottom are two buttons: 'Votar Pregunta' (blue) and 'Cancelar' (red). The bottom screenshot shows a confirmation dialog box with a yellow warning icon and the text 'REGISTRAR VOTO' and 'Confirme su voto por la opción: SI (1)'. It has 'CANCELAR' and 'CONFIRMAR' buttons. The background of the bottom screenshot is a blurred version of the top screenshot.

Ilustración 8. Menú de votaciones del aplicativo GoPH.

En cada asamblea hay anexos, pueden ser informes, citación de asamblea o incluso el formato de registro de poderes. Por esto, en el aplicativo se diseña la sección de informes, la cual tiene correspondencia con una tabla donde se encontrará el nombre del documento, la fecha en que se subió y el tamaño del mismo, para que los usuarios puedan acceder a ellos mediante un botón de ver documento (Ver **Ilustración 9**).



Ilustración 9. Informes - Anexos asamblea del aplicativo GoPH.

Para mayor comodidad del usuario se diseña una sección en los informes para ver el historial de votaciones que se han realizado, de esta forma el usuario podrá consultar sus votos. Esta sección cuenta en la parte superior con la cantidad de preguntas realizadas, y en el centro de la ventana cada una de las preguntas realizadas durante la asamblea, hace cuanto fue creada la pregunta y un botón para consultar el voto, mediante el cual se mostrará un popup con la opción por la cual votó (Ver **Ilustración 10**).



Ilustración 10. Informes - Historia votos del aplicativo GoPH.

Por último en la sección de informes se diseña una ventana donde los usuarios puedan ver los poderes con los que se encuentran registrados en la asamblea, en la parte superior de esta ventana se podrá ver el nombre de la persona con su coeficiente de participación, los poderes que tiene y el coeficiente de participación en la asamblea; y en la parte central de la ventana se puede observar cada uno de los poderes, con su respectivo nombre del propietario del inmueble, la dirección del inmueble y el coeficiente del predio (Ver **Ilustración 11**).

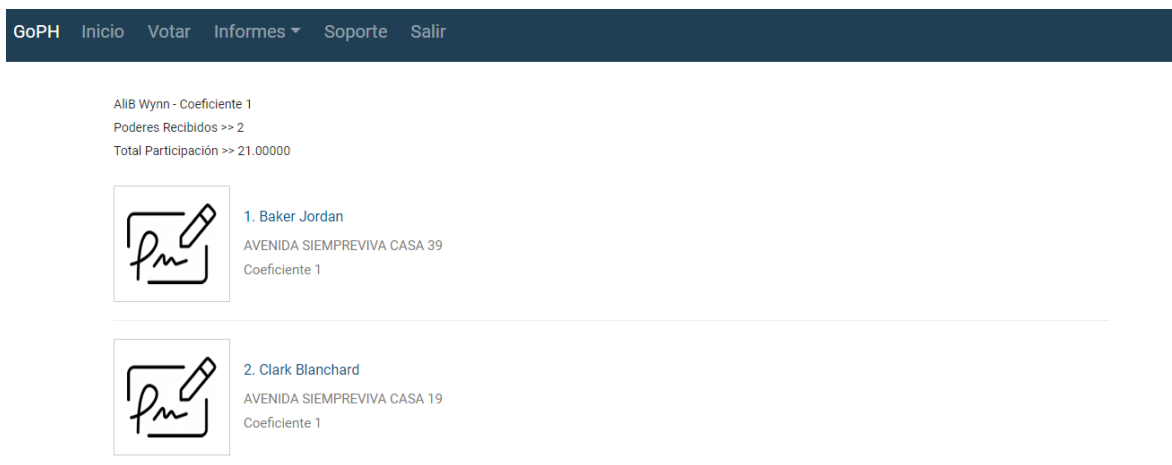


Ilustración 11. Informes - Mis poderes del aplicativo GoPH.

Dado que la mayoría de usuarios que utilizan el aplicativo GoPh son mayores a los 40 años, se diseñó una ventana para el soporte técnico, donde se podrá encontrar los números de soporte técnico en tarjetas individuales, estas tarjetas están compuestas por un icono de teléfono, otro de WhatsApp, el número de soporte técnico y un correo electrónico (Ver **Ilustración 12**).

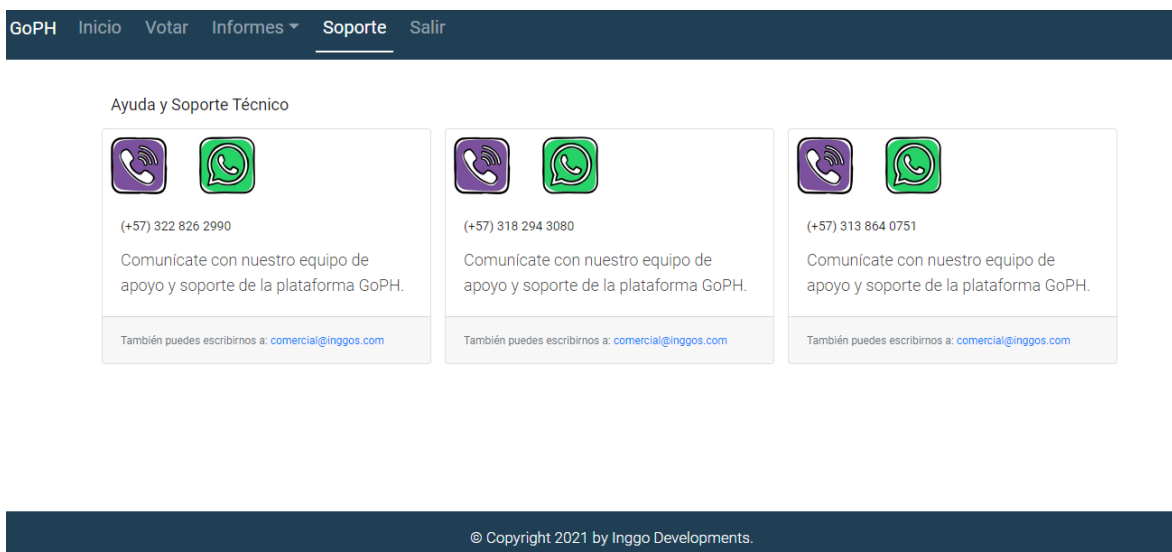


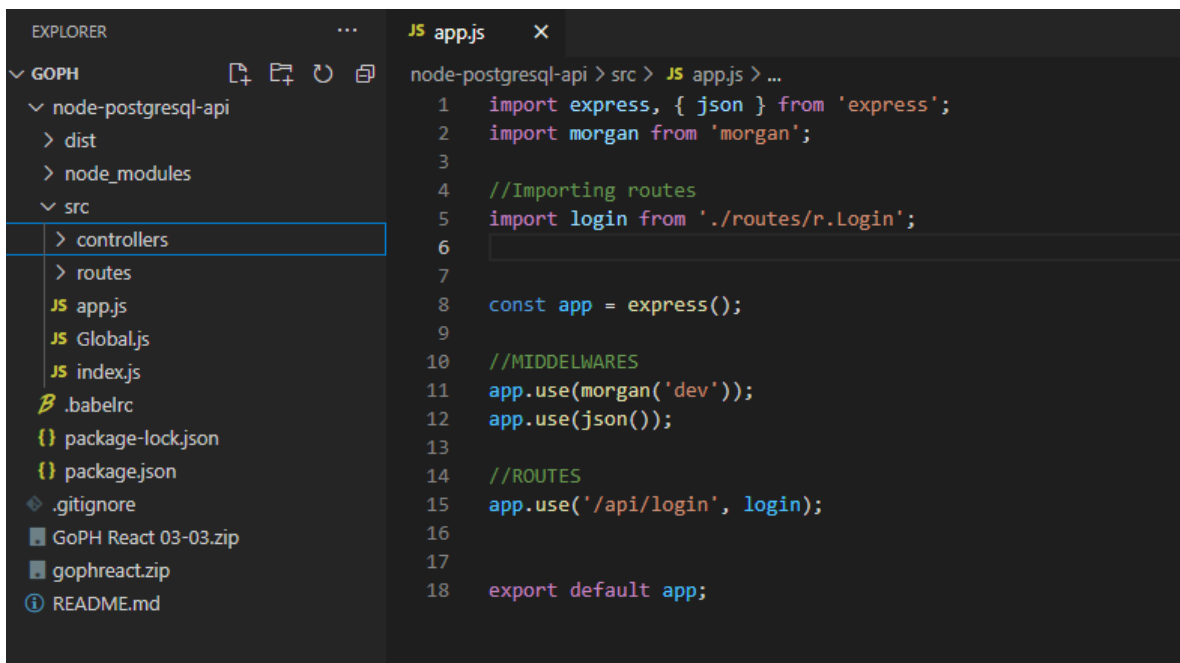
Ilustración 12. Soporte del aplicativo GoPH.

8.2. Desarrollar el Backend del aplicativo en Node.JS

Para iniciar la creación del Api Restfull con Node.JS es necesario primero realizar la instalación pertinente de Node.JS y npm, esta se descarga desde el sitio oficial de NodeJS.org; después de su instalación se crea una carpeta la cual albergará nuestra Api, al ingresar a nuestra carpeta desde una terminal se ejecutara el comando “*npm init -y*”, este lo que hace será crear un package.js con el nombre del proyecto, el autor, el número de la versión, etc; ya con esto se procede a instalar las librerías que serán requeridas para hacer la Apirestful, se instalan los módulos principales para nuestra backend express, unas librerías que nos permitan interactuar con nuestra base de datos en postgresql, un convertidor de código de JavaScript a JavaScript moderno, y por último un módulo llamado morgan el cual nos recibirá nuestras peticiones http.

Para instalar todo esto se corre el comando “*npm install express pg pg-hstore sequelize morgan @babel/polyfill*”. Ahora se instalarán los módulos secundarios de babel para esto se ejecuta el siguiente comando “*npm install --save-dev @babel/core @babel/cli @babel/preset-env*”, igualmente se instala la librería nodemon que permite no tener que ejecutar el código cada vez que se realice un cambio sino mantener el código ejecutando y actualizando solo con guardarlo, se instala con el siguiente comando “*npm install nodemon -D*” y “*npm install @babel/node -D*”.

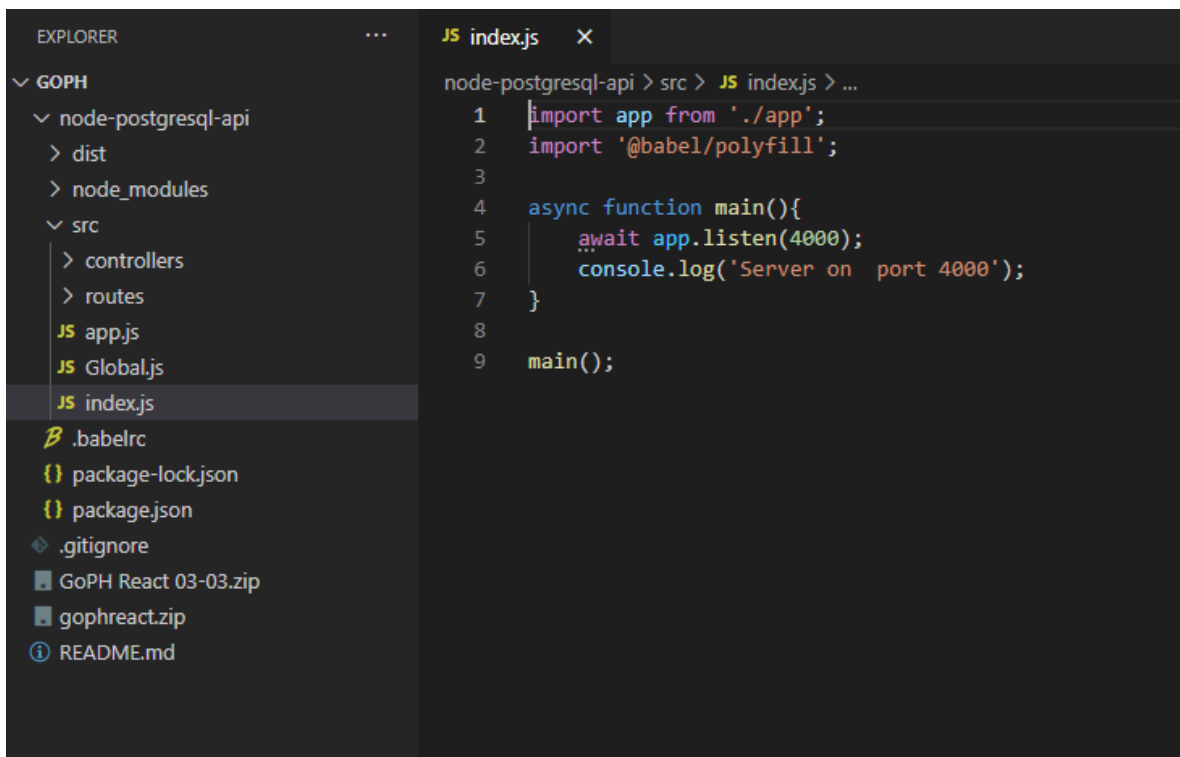
Posteriormente se crean las carpetas en la forma como estará distribuido en los archivos en el aplicativo (con ello se procura darle un poco más de estructura y orden al proyecto), para esto se crea una carpeta src, y dentro un index.js, app.js, una carpeta llamada *routes* y otra llamada *controllers*. El archivo index.js es quien estará encargado de ejecutar la configuración del servidor, el cual se encuentra localizado en app.js, se configura el archivo app.js, se empieza importando las librerías a utilizar como express, json y morgan, luego se importan los archivos *routes*, se configura *express* en app, se configuran los middlewares, se configuran los *routes* importados y se exporta el archivo configurado app (Ver **Ilustración 13**).



```
node-postgresql-api > src > JS app.js > ...
1  import express, { json } from 'express';
2  import morgan from 'morgan';
3
4  //Importing routes
5  import login from './routes/r.Login';
6
7
8  const app = express();
9
10 //MIDDELWARES
11 app.use(morgan('dev'));
12 app.use(json());
13
14 //ROUTES
15 app.use('/api/login', login);
16
17
18 export default app;
```

Ilustración 13. app.js Backend.

Con el archivo app.js configurado, se procede a configurar el archivo index, el cual solo cumplirá la función de escuchar y ejecutar el archivo app.js (Ver **Ilustración 14**).



```
node-postgresql-api > src > JS index.js > ...
1  import app from './app';
2  import '@babel/polyfill';
3
4  async function main(){
5      await app.listen(4000);
6      console.log('Server on port 4000');
7  }
8
9  main();
```

Ilustración 14. index.js Backend.

Al tener lista la configuración de estos dos archivos, index.js y app.js, se procede a crear las rutas, en la carpeta routes, donde estará todas las rutas de la api, estas rutas serán llamadas por medio de peticiones get, post, patch y put, y según también la ruta que traiga la petición. Para crear una ruta es necesario importar el módulo Route de express, instanciar este módulo en una constante y usarla para crear las rutas según la petición que se desee, la dirección que está escuchando y la función que realizará. Para esto será necesario llamar las funciones de los controladores en la carpeta controller, y estas serán llamadas según sea el tipo de petición y la ruta indicada; al final se debe exportar esta constante donde fue instalado el módulo Router de express (Ver **Ilustración 15**).

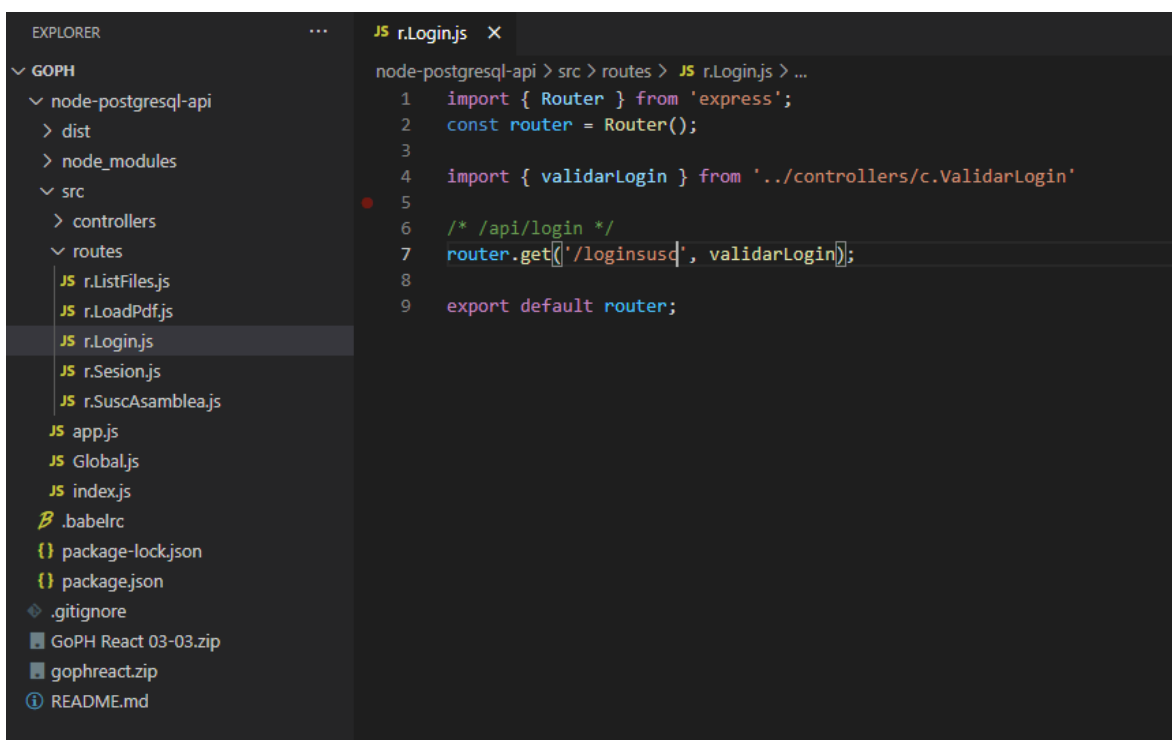


Ilustración 15. r.Login.js Backend.

Como se observa en la anterior ilustración, los archivos routers, solo son un intermediario entre las peticiones y las funciones que interactúan con la base de datos, archivos, etc. Para esto se crea la carpeta controllers en la cual están todos los controladores, funciones, e interacciones con los servidores locales, es allí, donde por medio de sequelize se conecta a la base de datos, se realizan los respectivos procesos, de consulta, eliminación, actualización o

inserción a la base de datos. De igual manera estos controladores responden a las peticiones con un archivo Json. Debido a la sensibilidad de datos que se manejan aquí no es posible mostrar como las anteriores configuraciones.

Para realizar las peticiones de prueba pertinentes se utiliza el programa “insomnia” el cual permite simular las peticiones http del navegador, de esta manera todos los controladores, y las rutas quedan verificadas y listas para ser consumidas por el front end.

8.3. Desarrollar la migración del Frontend a React JS

La migración del aplicativo se inicia descargando el framework React con el comando “*npm install -g npm@latest*”; posteriormente, se limpia la cache con el comando “*npm cache clean --force*”, esto para que nada que pueda afectar nuestro nuevo aplicativo quede en caché y, por último, se crea el proyecto con el comando “*npm install -g create-react-app*” donde se le coloca el nombre y la ubicación del archivo, de la misma manera como en el backend se instalan las librerías adicionales que se utilizarán, y se irán instalando.

En React JS cada vista se maneja por componentes individuales para tener todo organizado y separado, aunque todo parece estar junto, en realidad todo está dividido por los componentes. Se inicia una discusión con el equipo de trabajo para analizar la manera más adecuada de dividir el bosquejo realizado en componentes para optimizar código y realizar unas buenas prácticas de programación; ya tomada esta decisión se procede a crear los componentes pertinentes, donde en cada componente se empieza a colocar el html necesario (Ver **Ilustración 16**).

```

JS CopyRight.js X
gophreact > src > components > JS CopyRight.js > [0] default
1  import React, { Component } from 'react';
2
3  class Copyright extends Component {
4
5      render() {
6
7          return (
8              <div className="footer_bottom">
9                  <div className="container">
10                     <div className="row">
11                         <div className="col-12">
12                             <p id="crp" className="crp">© Copyright 2021 by Inggo Developments.</p>
13                         </div>
14                     </div>
15                 </div>
16             </div>
17         );
18     }
19 }
20
21 export default Copyright;

```

Ilustración 16. CopyRight.js Frontend.

Como se puede observar en la ilustración anterior, las importaciones de funciones, de librerías, módulos e imágenes a utilizar se realizan al comienzo de cada componente para el buen funcionamiento de este. Dentro de la clase extensión del método componente, se declara el state, que permite interactuar a las variables con las respuestas que lleguen a las peticiones, al finalizar estas declaraciones se continúa con los ciclos de vida del software, donde principalmente se instala el ciclo de vida “componentWillMount()” donde se inicializan las principales variables, se realizan las peticiones iniciales con las cuales se realizarán las primeras peticiones para realizar las validaciones correspondientes. Dentro de la clase creada a partir del método componente de React JS, se pueden declarar los métodos propios y funciones, a partir de funciones de flechas las cuales serán utilizadas según su necesidad. Para finalizar en el render va la parte gráfica, y parte de código lógico jsx, este interactúa directamente con los cambios en las variables permitiendo reaccionar al instante si hay algún cambio en alguna variable del state.

8.4. Implementar cifrado a cada petición que haga el Frontend al Backend

Al concluir con la migración y con cada una de las peticiones realizadas desde el frontend al backend, se procede a cifrar estas mismas para mayor seguridad, puesto que fueron usadas de forma plana para poder trabajar más fácilmente con ellas; para este ejercicio se utilizó un

cifrado *Aes modo cbc a 256 bits* adicionando un valor salt, esto para brindarle mayor seguridad.

Por temas de privacidad no se hace público la librería a utilizar y la forma de cifrado.

8.5. Pruebas de software migrado y salida a producción

Para sacar el software a producción fue necesario realizar las pruebas pertinentes sobre cada uno de los procesos, marcar asistencia, votar, validar asistencia, sesión, etc; cada una realizada.

Las pruebas del producto (Ver **Ilustración 17**) desarrollado por el equipo y a fin de estar seguro acerca de la funcionalidad, el rendimiento y la experiencia del usuario fueron desarrolladas de manera manual y automatizadas, se detallan a continuación:

Pruebas unitarias: Se realizaron pruebas unitarias a las funciones, procedimientos y módulos de la aplicación desarrollada, se aplicaron correcciones para obtener el comportamiento esperado.

Pruebas de integración: Se validó la integración de diferentes módulos juntos y se identificaron los errores y problemas relacionados con ellos.

Pruebas funcionales: Se comprobó la funcionalidad y la usabilidad para garantizar que las características del software se comportan según lo esperado, sin ningún problema, según las especificaciones de los requerimientos del software.

Pruebas no funcionales: Actualmente se están desarrollando las pruebas de carga al aplicativo, hasta el momento se han realizado seguimientos permanentes obteniendo los resultados esperados.

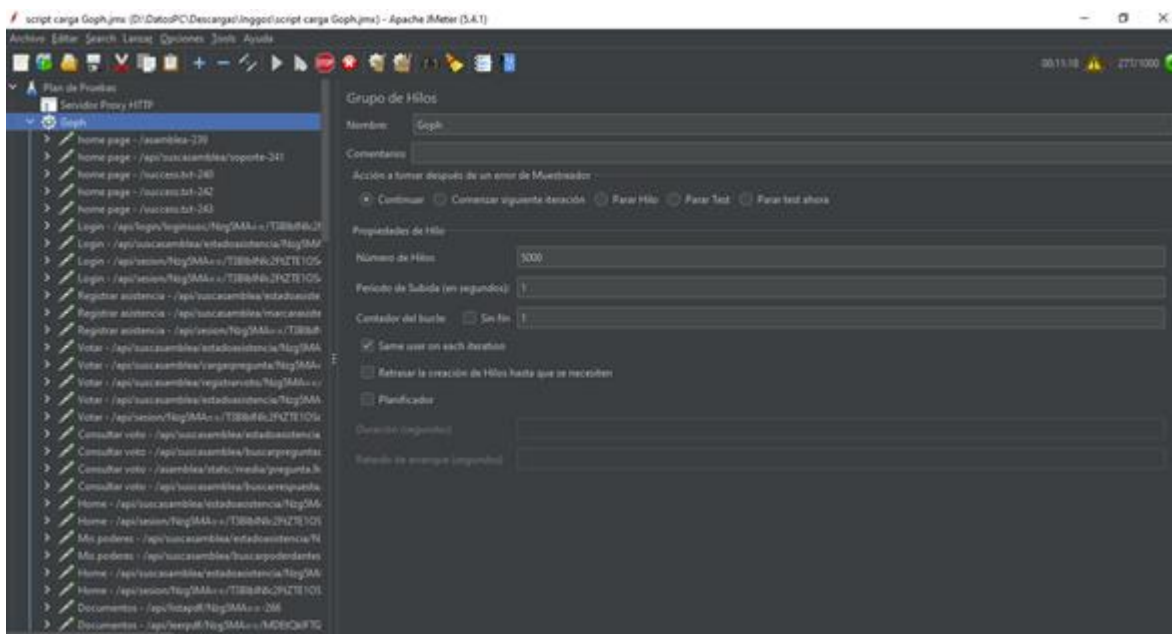


Ilustración 17. JMeter Hilos para realizar pruebas.

Como se puede observar en la anterior imagen se puede ver la cantidad de Hilos (pruebas a realizar), que se realizarán a cada script del aplicativo, debajo de esto el periodo de salida de cada uno y por último la cantidad de bucles a aplicar, que en este caso solo fue uno.

Por otro lado, están los resultados obtenidos que se pueden observar en las **Ilustración 18 y 19**, donde a mano derecha se puede observar una tabla donde está de primero el nombre del archivo al que se le hizo la prueba seguido de sus valores máximos y mínimos de tiempo, la media de cada ejecución.

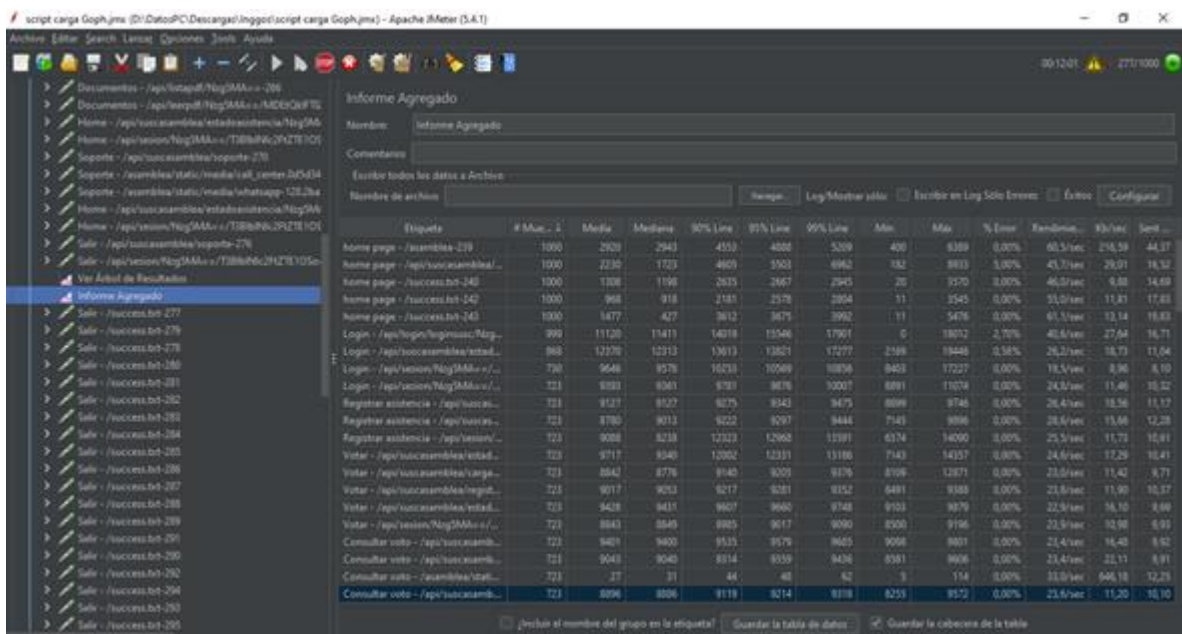


Ilustración 18. JMeter Informes Parte 1.

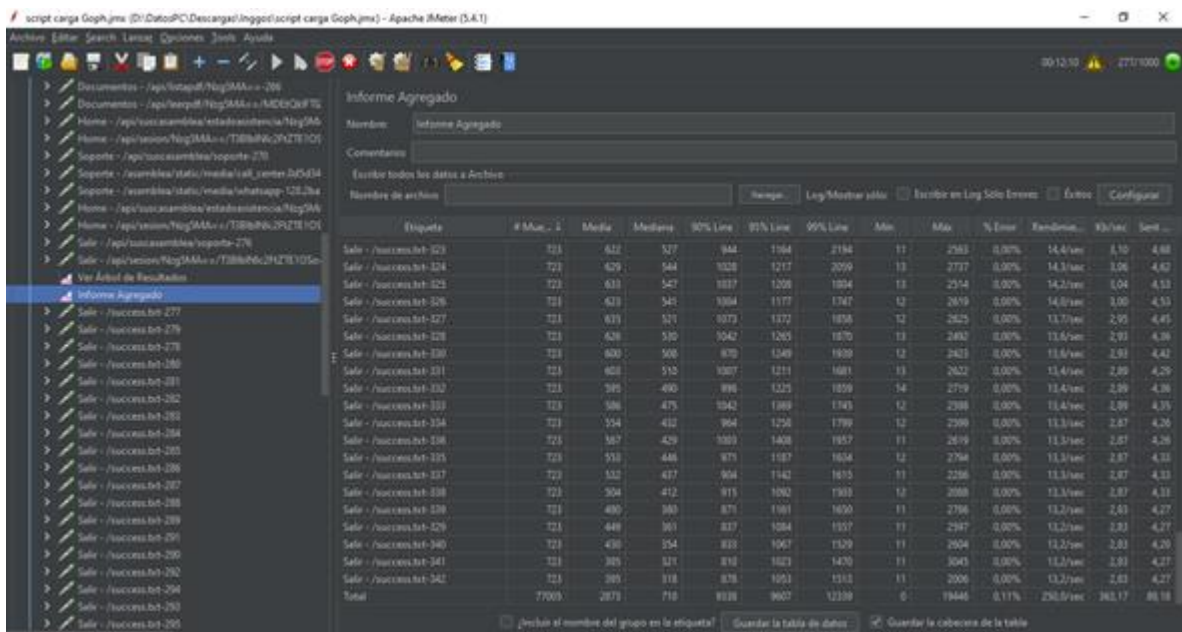


Ilustración 19. JMeter Informes Parte 2.

Con las pruebas realizadas, como se puede observar en la **Ilustración 18 y 19**, se puede iniciar la compilada y publicación de cada uno de los proyectos, para este caso en ambos softwares frontend y backend, se compila con el comando “npm run build” que nos genera las carpetas para publicar en producción.

Se requería el uso de otras herramientas para sacar a producción los softwares elaborados, donde fue necesario implementar PM2 que permite que el backend mantenga en línea, conocido como demonio, sin necesidad de errores en las peticiones se detenga el servicio o por un reinicio del servidor sea necesario volver a iniciarlo manualmente; por otro lado, fue necesario implementar NGINX que es otro administrador de servicios web aparte del IIS o APACHE, siendo este el principal para administrar dos servidores con el frontend para mantener un servicio balanceado.

Adicionalmente se creó un servidor virtualizado para realizar el balanceo de carga y que haga la función de repartidor, a los servicios viejos (Software administrativo y App) y los nuevos. Este nuevo servidor es instalado en Ubuntu, por su facilidad y eficiencia a la hora de funcionar de balanceador y administrador de páginas web con NGINX.

De esta manera se diseña que al realizar la petición a la página web, llega principalmente al servidor de Ubuntu este es el encargado de repartir estas peticiones entre los dos servidores que están destinados al servicio del frontend, desde allí se instancia también una regla que permita acceder a los servicios del aplicativo viejo, por último se instancia otro balanceador de carga adicional para impedir que un solo servidor de backend se recargue y no pueda prestar el servicio óptimo, así se balancea también entre dos servidores de backend.

8.6. Propuesta de un framework genérico para la migración de aplicativos webs

Durante el desarrollo de la práctica se realizó la migración sin seguir un marco de trabajo que indicara con claridad los pasos a seguir, situación que dificultó el proceso. Por tal motivo, se realizó como actividad adicional a este trabajo, una investigación sobre las metodologías y frameworks existentes en la literatura para realizar migración de aplicativos web similares al desarrollado en la práctica académica, sin embargo, no se encontró alguno que permitiera satisfacer las necesidades específicas de este proyecto.

Por lo anterior, se presentan en esta sección una propuesta de framework abstracto que podría ser utilizada por la empresa para futuras migraciones. La secuencia de pasos está dividida en tres secciones, cada una de las cuales contiene a su vez las actividades recomendadas para llevar a cabo la actividad.

1. Pre-migración

- a. Estudio de la tecnología actual: Se analizan las características de las tecnologías en las cuales está desarrollado el aplicativo a migrar, de tal manera que se tenga claridad sobre los aspectos a mejorar en la migración.
- b. Estudio de la nueva tecnología: Se realiza el estudio sobre la tecnología que se utiliza para la migración del aplicativo web, buscando suplir a través de estas, las deficiencias del aplicativo con la tecnología anterior.
- c. Planificación de la migración: Determinación de los diferentes procesos, artefactos, y actividades a desarrollar durante el proceso de migración.
- d. Diseño del sistema: Se centra en el diseño del nuevo aplicativo web, considerando tanto las interfaces gráficas (ventanas, funciones, respuestas, informes) como la arquitectura de software sobre la cual se realizará el despliegue.

2. Migración

- a. Codificación y aplicación de procesos: Implementación de los bosquejos realizados a través de código (en las nuevas tecnologías), ejecutando paso a paso cada proceso de migración determinado en el literal c de la Pre-migración.
- b. Versionamiento: Proceso de gestión de la configuración de los nuevos artefactos desarrollados.

3. Post-migración

- a. Testeo: Se hacen las pruebas pertinentes del aplicativo web y la verificación de cada uno de sus funcionamientos y respuestas. En este paso pueden ser implementadas pruebas unitarias, pruebas de integración o pruebas funcionales, entre otras.
- a. Documentación: Se hace el manual de usuario y la documentación respectiva del aplicativo para su mantenimiento.
- b. Lanzamiento a producción: Se realiza la publicación del aplicativo web y la actualización de los servidores (si es requerido por la tecnología utilizada).

9. CONCLUSIONES

Este trabajo presenta la migración exitosa del aplicativo GoPH al Framework React y al Backend Node Js. En la realización del trabajo se logró culminar con éxito las tareas asignadas, adquiriendo nuevas experiencia y nuevos conocimientos propios de la carrera.

Respecto al trabajo se concluye lo siguiente:

- Se desarrolla el backend en Node.JS permitiendo conectarse a una base de datos en PostgreSQL.
- Se implementaron tanto las peticiones get como las peticiones post, put, en el desarrollo de la migración, entre el cliente y el servidor.
- La fusión de la plantilla creada en Html5 a React JS, fue satisfactoria, descomponiendo y creando los componentes necesarios para su respectivo funcionamiento.
- Al aplicar la lógica requerida se logra la migración respectiva con el correcto funcionamiento.
- En las pruebas realizadas se logró determinar que con las pruebas manuales y auditadas a fondo se pudieron identificar errores antes de lanzar el producto. Consideramos necesario que los problemas no se repliquen en el entorno de producción.
- Se propuso un modelo de trabajo (framework) para futuras migraciones de aplicativos web, de tal manera que se facilite tanto la codificación y documentación del sistema a migrar, como los procesos de gestión relacionados, permitiendo disminución en tiempo y personal que pueden ser significativos para la empresa en términos de eficiencia.

10. RECOMENDACIONES

- En el desarrollo de la práctica, se encontró algunos inconvenientes a la hora de estar contra tiempo para la entrega ya que a veces las pruebas no eran las más adecuadas, por lo cual se sugiere planear con un mayor plazo de tiempo una próxima migración.
- Se recomienda la implementación de una metodología ágil, para el versionamiento y el desarrollo de futuras aplicaciones.
- Para el backup de las versiones desarrolladas e implementadas se recomienda utilizar un sistema de backups automático, sin necesidad de depender del equipo de trabajo lo suba al repositorio manualmente.
- Para una migración rápida y adecuada es necesario realizar una contextualización previa de los temas a intervenir, para tener claridad de lo que se desarrollará y objetividad durante la planificación y ejecución del proceso.
- Para futuras migraciones de sistemas web, se recomienda utilizar un modelo de trabajo como el propuesto durante las prácticas, de tal manera que se realice un proceso planificado, ordenado y eficiente, evitando inconvenientes relacionados con la no documentación, cambios no percibidos durante la migración o artefactos faltantes en la migración.

11. REFERENCIAS

- [1] R. S. Pressman, Ingeniería de software un enfoque práctico, Edición 7, México: McGRAW-HILL INTERAMERICANA EDITORES, 2010.
- [2] IEEE “IEEE 802 93a”, IEEE, 802, 2019.
- [3] A. Toro, L. Cardona, “ESTADO DEL ARTE DE LA INGENIERÍA DEL SOFTWARE EN EL ÁMBITO NACIONAL E INTERNACIONAL DE ACUERDO A ORGANIZACIONES QUE TRATAN LA DISCIPLINA”, Universidad Catolica de Pereira, Colombia, Pereira, 2010.
- [4] A. A. Espinoza, "Manual para elegir una metodología de desarrollo de software dentro de un proyecto informático," thesis, facultad de ingeniería, Universidad de Piura, Piura, Perú, 1997.
- [5] J. L. Enríquez, et al. “METODOLOGÍA DE DESARROLLO DE SOFTWARE,” Universidad Católica los Ángeles Chimbote, Chimbote, Perú, v. 1, 2017.
- [6] E. G., Maida, J. Pacienza, “Metodologías de desarrollo de software” Tesis de Licenciatura en Sistemas y Computación. Facultad de Química e Ingeniería. Universidad Católica Argentina, 2015, octubre 30 2021, Online: “<http://bibliotecadigital.uca.edu.ar/repositorio/tesis/metodologias-desarrollo-software.pdf>”.
- [7] ISO/IEC “ISO 12207-1”, ISO, 12207, 1995.
- [8] V. Robles, (2019, Sept 3) “Master en Frameworks JavaScript: Aprende Angular, React, Vue”, enero 2021, Online: “<https://www.udemy.com/course/master-en-frameworks-javascript-aprende-angular-react-vue-js/>”.
- [9] IEEE “IEEE 1074”, IEEE, 1074, 1989.
- [10] P. Domínguez, (2018, Nov 5) “Gestiona tu proyecto de desarrollo” mayo 5 2021, Online: “<https://openclassrooms.com/en/courses/4309151-gestiona-tu-proyecto-de-desarrollo/4538221-en-que-consiste-el-modelo-en-cascada>”.
- [11] E., Rosa, Introducción a la teoría de la arquitectura, pp. 17, Primera edición, México: RED TERCER MILENIO, 2012.
- [12] IEEE “IEEE 1471”, IEEE, 1471, 2000.

- [13] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Noed, J. Stafford, “Documenting software architectures, views and beyond” segunda edición, E.E.U.U.: Addison-Wesley Professional, 2010.
- [14] S. J., López, (2008, Feb 22) “Modelo cliente servidor”, Online: “<https://redespomactividad.weebly.com/modelo-cliente-servidor.html>”.
- [15] NPMJS, (2019, Dec 9) “Node Package Manager”, mayo 5 2021, Online: “<https://docs.npmjs.com/about-npm>”.
- [16] Node.JS, (2014, Oct 20) “Node JS”, mayo 5 2021, Online: “<https://nodejs.org/en/>”.
- [17] ReactJS, (2019, Feb 20) “React JS”, mayo 5 2021, Online: “<https://reactjs.org/>”.
- [18] HTML, (2012, May 23) “HTML”, mayo 5 2021, Online: “https://www.w3schools.com/html/html_intro.asp”.
- [19] CSS, (2007, Dec 8) “CSS”, mayo 5 2021, Online: “https://www.w3schools.com/css/css_intro.asp”.
- [20] Bootstrap, (2020, Dec 7) “Bootstrap”, mayo 5 2021, Online: “<https://getbootstrap.com/docs/5.0/getting-started/introduction/>”.
- [21] PostgreSQL, (2018, Abr 18) “PostgreSQL”, mayo 5 2021, Online: “<https://www.postgresql.org/>”.
- [22] SQL Manager For PostgreSQL, (2021, Abr 9) “SQL Manager For PostgreSQL”, mayo 5 2021, Online: “<https://www.sqlmanager.net/products/postgresql/manager>”.
- [23] JMeter, (2011, Nov 8) “JMeter”, mayo 5 2021, Online: “<https://jmeter.apache.org/>”.
- [24] Nginx, (2021, Oct 28) “Nginx”, noviembre 5 2021, Online: “<https://www.nginx.com/>”.