ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Optimización y expansión de módulos del Sistema PREC para edificios Inteligentes

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniero en Ciencias de la Computación

Presentado por:

Sebastian Alejandro Mendoza Ramírez

GUAYAQUIL - ECUADOR

Año: 2022

DEDICATORIA

A Dios, por haberme permitido llegar a este punto y haberme dado salud para alcanzar mis objetivos.

A mamá, por su amor y apoyo incondicional. Gracias por ser esa amiga que me ha ayudado a crecer, acompañándome siempre en mis logros y en mis derrotas. Sin ti no estaría hoy aquí.

A mis abuelas, por sus oraciones y fe incondicional. Gracias por creer en mí.

A mis hermanos, que con cariño me han enseñado a construir mi caminar. Gracias por preocuparse por su hermano menor y compartir sus vidas conmigo, pero, sobre todo, gracias por estar en otro momento tan importante de mi vida.

A mi familia, por su aliento y apoyo incondicional.

A papá, por su amor, cariño y complicidad. Llevo tu recuerdo tatuado en el corazón...

AGRADECIMIENTOS

Al Ph.D Federico Domínguez, por sus correcciones, sugerencias y apoyo en el desarrollo de esta tesis.

Al Mag. Erick Lavid Cedeño, por sus comentarios y guianza a lo largo de este camino.

Al Ph.D José Córdova y al Ing. Alberto Cruz, por brindarme la oportunidad de formar parte de la iniciativa PREC.

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, me(nos) corresponde conforme al reglamento de propiedad intelectual de la institución; Sebastian Alejandro Mendoza Ramírez y doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual”

Sebastian Alejandro Mendoza Ramírez

EVALUADORES

Firmado electrónicamente por:
ERICK VICENTE LAVID CEDENO

Mag. Erick Lavid Cedeño

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
FEDERICO XAVIER DOMINGUEZ BONINI

Ph.D Federico Domínguez Bonini

PROFESOR TUTOR

RESUMEN

El presente proyecto integrador tiene como principal objetivo la optimización del sistema PREC, el cual presenta problemas de rendimiento y no cubre actualmente las necesidades de sus usuarios de información relevante y de calidad. Este objetivo fue alcanzado por medio de un proceso iterativo de análisis y reingeniería, obteniendo una menor complejidad en los algoritmos encargados del funcionamiento del sistema a través de estrategias de optimización aprendidas a lo largo de la carrera. Adicionalmente, se implementaron nuevos módulos de reportería, tras la elicitación de requerimientos con usuarios del sistema, recopilando sus puntos de dolor y necesidades. Los resultados obtenidos demostraron ser sobresalientes, con mejoras en los tiempos de respuesta presentes en el sistema de hasta un 99.77% y la implementación de un módulo de reportería de consumos eléctricos dinámico, capaz de establecer comparativas entre zonas y el campus, las cuales son presentadas de forma gráfica, tabulada y exportable para la comodidad del usuario. De esta forma, y según la retroalimentación del cliente y los usuarios, se concluyó que las estrategias aplicadas fueron apropiadas para el estado inicial del sistema, y que los módulos implementados se encuentran alineados a las necesidades de los usuarios.

Palabras Clave: Optimización, Tiempo de Respuesta, Reingeniería, Rendimiento.

ABSTRACT

The main purpose behind this project is the optimization of the PREC system, which presents performance issues and fails to cover the users' needs for both relevant and quality information. This goal was achieved through an iterative process involving the system's analysis and reengineering, resulting in a lower complexity for the algorithms responsible of the system's underperformance. In addition, new dashboards for electrical power consumption were implemented under users' specifications, covering both their needs and desires. The final results proved to be outstanding, with response time reductions of up to 99.77% and the new implemented dashboards receiving positive feedback from the end users. The conclusions reached towards the end of this project are that the strategies applied to the different scenarios proved to be both appropriate and effective, and that the newly implemented modules are aligned to both the project and user needs, due to an effective requirement elicitation project, focused on the final user as its main source of information and feedback.

Keywords: *Optimization, Response Time, Reengineering, Performace.*

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS

ESPOL	Escuela Superior Politécnica del Litoral
PREC	Power Reduction for Energy Consumption
IOT	Internet of Things

SIMBOLOGÍA

s	Segundos
ms	Milisegundos

CAPÍTULO 1

1. INTRODUCCIÓN

En la actualidad, los sistemas en general experimentan constantemente un incremento en los volúmenes de información, indiferentemente del fin que esta posea. Esto es debido a que, desde la segunda década de los años 2000, se han evidenciado grandes avances en temas de conectividad, banda ancha, movilidad, etc. Asimismo, ha incrementado de forma exponencial el número de usuarios en los sistemas computacionales, producto de la reducción en costes de conexión, dispositivos inteligentes y el incremento en las habilidades y destrezas tecnológicas de las poblaciones [1].

Un claro ejemplo de esto es el Internet de las Cosas, o IOT según sus siglas en inglés, el cual es paradigma tecnológico, el cual consiste en una red de dispositivos capaces de interactuar unos con otros, por medio de señales, mensajes, entre otras[2]. Una de las aplicaciones de este paradigma son los edificios inteligentes, en los cuales se instala una multitud de dispositivos con la finalidad de recopilar información sobre este, automatizar procesos manuales que ocurren dentro del mismo, etc.

Esta recopilación de información se realiza a través de sensores, los cuales se encuentran constantemente enviando información a un sistema administrador, el cual permite la recopilación y filtrado de esta para mostrar a los usuarios finales.

Actualmente, la Escuela Superior Politécnica del Litoral se encuentra en proceso de implementación de un Sistema para Edificios Inteligentes, a esta iniciativa se la denominó PREC, de las siglas en inglés para Power Reduction for Energy Consumption.

El proyecto Power Reduction for Energy Consumption (PREC) tiene sus orígenes en el concurso i3Lab, donde nace como propuesta de un grupo de docentes de la Escuela Superior Politécnica del Litoral, con el objetivo de reducir el desperdicio de un valioso recurso como lo es la energía eléctrica. Esto, por medio de la implementación de diferentes tipos de sensores a lo largo del campus, para la recopilación de valiosa

información referente a consumos eléctricos, uso de ambientes, etc. Así como una plataforma capaz de permitir a los usuarios acceder a esta información, a través de gráficos, reportes, etc. Actualmente, se cuenta con una versión preliminar de la plataforma, aunque esta cuenta con ciertas limitaciones.

1.1 Descripción del problema

La versión preliminar del sistema PREC cuenta con módulos de Visualización y Comparativa de Datos de consumo eléctrico en diferentes zonas y edificaciones del campus universitario. Esta versión del sistema cuenta también con módulos de Facturación Electrónica, Gestión de Cargas Eléctricas y en último lugar un módulo de Datos en Tiempo Real. Sin embargo, algunos de estos módulos presentan grandes tiempos de carga, tiempo durante el cual el usuario no puede hacer uso del sistema. Adicionalmente, el sistema actual no cubre de forma satisfactoria las necesidades de los usuarios en términos de acceso a información, puesto que carece de módulos para la consulta y visualización de consumos globales y zonales de energía eléctrica de forma anual. En último lugar, el sistema carece de cobertura para la totalidad de sensores disponibles, puesto que carece de módulos para la consulta y visualización de datos de sensores ajenos al consumo eléctrico, los cuales han sido ya adquiridos y se encuentran en proceso de instalación.

1.2 Justificación del problema

Conforme aumenta el volumen de datos generados, aumenta también el volumen de datos con el que los sistemas han de estar en capacitad de lidiar. El incremento del tamaño de la data disponible tiene un impacto directo en los tiempos de consulta y procesamiento de la información. Esto, en muchas ocasiones, es solventado por medio del escalado de los sistemas, ya sea en dirección vertical u horizontal. La primera, supone la mejora a nivel de hardware, mejorando componentes como el procesador o incrementado la memoria disponible. Esto es una solución eficaz en muchos casos, sin embargo, es poco viable en el largo plazo, debido a los altos costos que puede llegar a significar. El escalado horizontal, por otra parte, hace referencia a levantar instancias

adicionales de los sistemas, a fin de que puedan procesar un número mayor de solicitudes en el mismo intervalo de tiempo, reduciendo la carga a la que se ve sometida cada instancia y así mejorando tiempos de respuesta. Sin embargo, esta es una opción válida cuando el problema consiste en el tráfico de la aplicación. Existe una tercera posibilidad, y es que, puede darse el caso que los procesos llevados a cabo en los sistemas sean poco eficientes, producto de una pobre planificación o ejecución en el diseño e implementación de las soluciones informáticas.

En el caso del Sistema PREC, es necesaria la revisión de los componentes Front y BackEnd del sistema, a fin de identificar las posibles causas de los altos tiempos de carga que manifiesta el sistema en su versión preliminar.

Adicionalmente, con miras a la reducción del consumo de energía eléctrica, es necesaria la recopilación información de calidad, que permita tener una visión acertada de la realidad. Es por esto que se requiere la implementación de sensores adicionales alrededor del campus, de forma que se pueda contar con información rica y variada para la detección de consumos que representan un malgasto de energía, esto a cargo del equipo de telemática del proyecto. Esto a su vez representará un incremento en el flujo de datos actual, por lo que la revisión de los componentes adquiere importancia adicional. De esta necesidad de nuevos y variados sensores, se desprende además la necesidad de poder acceder a la información recopilada por estos. Es por esto, que es importante el desarrollo de nuevos módulos en el Sistema PREC, módulos que permitan la visualización de las mediciones efectuadas por estos nuevos sensores.

1.3 Objetivos

1.3.1 Objetivo General

Optimización y Expansión del Sistema PREC, por medio de la identificación y corrección de fallas conceptuales en el diseño de consultas.

1.3.2 Objetivos Específicos

- Reducción de tiempos de carga en las consultas de consumos eléctricos del módulo Muestras

- Reducción de tiempos de carga en módulos de consulta de sensores
- Implementación de un módulo para la consulta y comparativa de consumos anuales globales y por zonas.
- Implementación de cambios en el diseño de interfaz gráfica en módulos del sistema

1.4 Marco Teórico

1.4.1 Stack en Aplicaciones IOT

En la actualidad, y en contra de la definición clásica de IOT, muchos de los dispositivos IOT de una arquitectura no se encuentran directamente interconectados entre sí, sino que se encuentran conectados a servicios, los cuales actúan como interfaces, permitiendo a otros dispositivos el acceso o envío de información, incluso llegando a servir de información a usuarios finales de los sistemas (humanos) [3].
Para el desarrollo de todas las capas involucradas en una arquitectura IOT, se emplea un conjunto de herramientas, al que se denomina “stack”. En el “stack” se incluyen las bases de datos, entornos de ejecución y frameworks, empleados en el desarrollo de la solución.

1.4.2 Bases de Datos

A la hora de construir una arquitectura de IOT, un factor crucial en el desempeño de la solución es el motor de base de datos empleado, puesto que este es el responsable de almacenar y extraer la información con la cual se va a trabajar. Es por esto que, la elección de un tipo de base de dato no es trivial y ha de ser realizada con cautela, teniendo en consideración las necesidades y requerimientos del sistema [4].
Actualmente, las bases de datos se clasifican en dos macro categorías, siendo estas Bases de Datos Relacionales y las Bases de Datos No Relacionales.

A Bases de Datos Relacionales

Las bases de datos relacionales, como su nombre sugiere, funcionan estableciendo relaciones, entre esquemas predefinidos que corresponden a los datos que modelan las

entidades intervinientes en el sistema. Las bases de datos relacionales hacen uso del Lenguaje de Consultas Estructurado (SQL), el cual posee las ventajas de ser flexible, robusto y con un rendimiento comprobado. Sin embargo, poseen la desventaja de que los esquemas definidos son poco flexibles, por lo que se han de crear varios esquemas para poder cubrir todas las necesidades o variedades de una entidad general, a diferencia de las Bases de Datos No Relacionales [4].

B Bases de Datos No Relacionales

Las Bases de Datos No Relacionales, en contraparte a las relacionales, no hacen uso de esquemas, por lo que funcionan sin necesidad de establecer relaciones entre sus colecciones, por lo que no es poco frecuente que estas sean usadas en proyectos de IOT, ya que permiten unificar las mediciones de diversos tipos de sensores bajo una misma colección [4].

C Comparativa Bases de Datos

Como se concluye en [4], cada tipo de base de datos tiene ventajas y desventajas de su uso en proyectos de IOT, por lo que, la elección de una base de datos apropiada depende de la clase de uso que se destine para esta, así como de la infraestructura disponible.
Rautmare y Bhalerao establecen que las Bases de Datos Relacionales trabajan más frecuentemente con un escalado vertical, mientras que las Bases de Datos No Relacionales trabajan con un escalado horizontal, implicando la replicación de las colecciones en múltiples nodos, los cuales han de estar sincronizados a fin de mantener la consistencia de los datos [4].
Esto quiere decir que, implementaciones de arquitecturas IOT en las que, por alguna razón, no sea posible el escalado horizontal, podrían hacer uso de bases de datos relacionales y viceversa.
Los autores en su publicación “MySQL and NoSQL database comparison for IoT application” realizan, además, una comparativa de rendimiento para bases relacionales y no relacionales, empleando ambas en el mismo proyecto donde se trabajó con sensores de temperatura, humedad, humedad de suelo y nivel de agua. En sus resultados, se puede apreciar un mejor rendimiento de las bases de datos relacionales en las consultas con condiciones de alto uso de CPU, considerando hasta un máximo de 10 hilos de CPU

en el caso de estudio.

Las bases de datos no relacionales ganan, sin embargo, en la inserción de datos en condiciones de alto uso de CPU, al presentar menores latencias que las bases relacionales.

En las consultas con grandes números de registros, las bases de datos relacionales obtienen nuevamente una ventaja por sobre su contraparte, al presentar menores tiempos de respuesta y con una curva más suave, es decir, con menos varianza y picos en los tiempos de respuesta. En lo que respecta a las inserciones de datos con grandes números de registros, las bases de datos no relacionales superan a las relacionales, obteniendo tiempos de respuesta más bajos en la mayoría de los casos, aunque, presentan picos y una curva menos suave en relación a la curva de las bases de datos relacionales, las cuales evidencian consistencia en los tiempos de inserción.

Los resultados expuestos dejan en claro que, se puede esperar un rendimiento superior de las bases de datos relacionales en Proyectos IOT que presenten un alto volumen de consultas, recopilando y procesando grandes volúmenes de registros. Por otra parte, las bases de datos no relacionales supondrían un mejor rendimiento en casos de Proyectos un gran afluente de datos, representado en un alto número de sensores instalados y transmitiendo información constantemente.

1.4.3 Ambiente de Ejecución

Los ambientes de ejecución hacen referencia al ambiente sobre el cual un programa o aplicación opera. En la actualidad, existen muchos ambientes de ejecución, los cuales son propios de las tecnologías y lenguajes empleados en el desarrollo de un sistema o aplicación. Por ejemplo, las aplicaciones escritas en Java corren sobre el Java Runtime Environment, o JRE por sus siglas. Otro ambiente de ejecución, el cual ha adquirido gran relevancia en tiempos recientes es Node.Js, el cual es un ambiente de ejecución para JavaScript que permite el desarrollo de aplicaciones de diversa índole, como lo son servidores o aplicaciones web [5].

1.4.4 Node.Js

Node.Js, también llamado simplemente Node, funciona sobre el motor V8, desarrollado por Google. Gracias a esto, Node.Js cuenta con una baja latencia y un alto rendimiento. Node cuenta también con la ventaja de ejecutarse de forma ininterrumpida sobre un hilo de procesador, desencadenando la ejecución de código por medio de eventos, lo cual es más eficiente y genera un menor uso de recursos que la creación de nuevos hilos como hacen otras tecnologías comúnmente empleadas en el desarrollo de servidores web [5].

1.4.5 Express.Js

Para el desarrollo de servidores web, Node.Js es comúnmente usado en conjunto con Frameworks, los cuales establecen una guía para el desarrollo de aplicaciones con fines específicos. Para el desarrollo de servidores web, una de las opciones más populares es el framework Express.Js, el cual provee un set de funcionalidades destinadas a la creación de servicios web de forma simple y robusta [6], brindando al desarrollador facilidades para el manejo y enrutamiento de solicitudes HTTP.

1.4.6 Angular

Angular es un framework para el desarrollo de aplicaciones web desarrollado y mantenido por Google y la comunidad, al ser de código abierto [3]. Este framework presenta al desarrollador con una implementación base, basada en el Modelo Vista Controlador [3], lo cual facilita el desarrollo al contar con un ambiente modular, separado por componentes, los cuales pueden ser exportados e importados dentro de otros para agilizar el desarrollo y fomentar el orden y separación de responsabilidades en el proyecto.

1.4.7 Stack en el sistema PREC

Actualmente, la plataforma PREC cuenta con las siguientes tecnologías en su stack:

1. MySQl: Motor de Base de Datos Relacional.
2. Express.Js: Framework para el desarrollo del servidor web (BackEnd)
3. Angular: Framework para el desarrollo de aplicaciones web (FrontEnd)

4. Node.Js: Ambiente de Ejecución JavaScript. Sobre este se ejecuta el BackEnd.

CAPÍTULO 2

2. METODOLOGÍA

2.1 Requerimientos

Antes de proceder con la definición del proceso metodológico de este proyecto, es necesario conocer las necesidades y requerimientos del cliente, a fin de poder evaluar y determinar la prioridad y viabilidad de estos. De esto se tiene lo siguiente:

2.1.1 Detalle de Requerimientos

Tras el proceso de levantamiento de requerimientos en conjunto con el cliente y administrador del sistema PREC se establecieron los siguientes requerimientos iniciales:

1. Se desea la revisión y reducción de tiempos de respuesta de los métodos de consultas en los módulos de Muestras y Sensores en Tiempo Real.
2. Se desea la implementación de cambios en la interfaz de usuario del sistema.
3. Se desea la implementación de un módulo para la consulta y comparativa de consumos por zona y a nivel ESPOL de forma anual.
4. Se desea poder realizar consultas sobre las mediciones efectuadas por los sensores de CO2.
5. Se desea poder realizar consultas sobre las mediciones efectuadas por los sensores de Humedad y Temperatura.
6. Se desea poder realizar consultas sobre las mediciones efectuadas por los sensores de Energía.

7. Se desea poder realizar consultas sobre las mediciones efectuadas por los sensores de Control AC.

8. Se desea poder realizar consultas sobre las mediciones efectuadas por los sensores de Ocupancia.

2.1.2 Beneficios y Objetivos de Requerimientos

Se espera que la implementación de los requerimientos antes mostrados tenga los siguientes beneficios u objetivos para con los usuarios del sistema PREC:

- Se reducen los tiempos de espera de los usuarios en la consulta de consumos eléctricos, acercando los tiempos de carga a los de las demás consultas disponibles dentro del sistema.

- Se identifican falencias en el diseño de consultas y los algoritmos que las realizan, dando oportunidad a mejoras.

- Se reducen los tiempos de espera de los usuarios para la consulta del estado de los sensores registrados en el sistema.

- Se permite al usuario el acceso a información referente a las mediciones realizadas por los diferentes tipos de sensores.

2.1.3 Análisis de Requerimientos

En esta sección de profundizará en la viabilidad de los requerimientos levantados, evidenciada en una propuesta de solución factible que se adapte al estado actual del sistema. Se explorarán tambien, las causas de la inviabilidad de algunos de los requerimientos.

A Soluciones Propuestas

Luego de evaluar el estado actual del sistema, se tienen las siguientes propuestas de solución para los siguientes requerimientos:

- Requerimiento 1: Documentación y Revisión de los Servicios de Back y FrontEnd involucrados en la recopilación de información mostrada al usuario en el módulo de Muestras. Tras la documentación, se realizará el monitoreo de los tiempos de respuesta de los servicios, a fin de identificar endpoints con altos tiempos de respuesta y oportunidades de mejora. Posteriormente, se realizará una revisión de los endpoints identificados, analizando la complejidad del algoritmo encargado de la consulta a fin de identificar posibles limitaciones en su implementación y así poder someterlo a un proceso de reingeniería, donde la complejidad del algoritmo se vea reducida.
- Requerimiento 2: Realizar, en conjunto con el cliente, la identificación de cambios a ser implementados a nivel de interfaz gráfica en el sistema PREC, esto producto del intercambio de información con los usuarios finales del sistema.
- Requerimiento 3: Realizar, en conjunto con los usuarios finales del sistema, el levantamiento de especificaciones para el nuevo módulo a ser implementado, a fin de que este cubra de forma satisfactoria los deseos y necesidades de los usuarios involucrados.

B Requerimientos No Realizables

Tras la evaluación del estado del sistema antes mencionada, se tienen los siguientes requerimientos no viables:

- Requerimiento 4
- Requerimiento 5
- Requerimiento 6
- Requerimiento 7
- Requerimiento 8

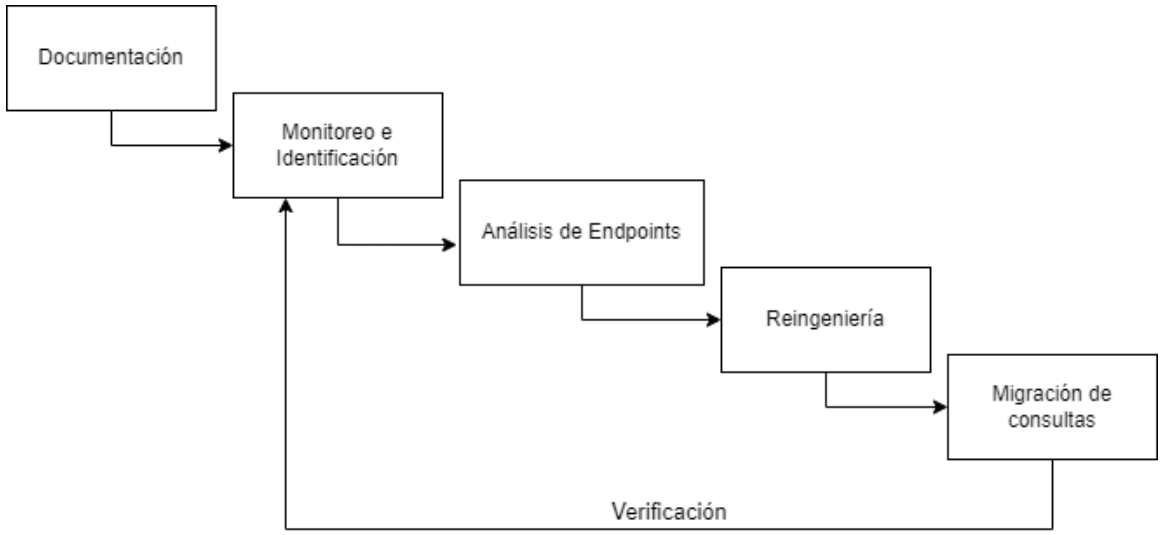
Los requerimientos mencionados no son realizables durante el periodo establecido como duración de este proyecto, puesto que los sensores en cuestión no se encuentran implementados a lo largo del campus, y no se cuenta con la infraestructura requerida para su conexión, monitoreo, registro y consultas. Es decir, estos sensores no se encuentran

actualmente instalados alrededor del campus, por lo que se carece de los esquemas y datos necesarios para la planificación e implementación de los módulos y sus respectivos endpoints.

2.2 Proceso Metodológico

En esta sección, se detallará el proceso metodológico de este trabajo, el cual es iterativo. Dentro de este proceso han sido definidos cinco pasos, que abarcan la documentación, monitoreo, análisis, reingeniería y migración de las consultas y módulos presentes en el sistema PREC, como muestra la Figura 2.1:

Figura 2.1: Representación visual del proceso metodológico. [Autoría Propia]



2.2.1 Documentación

Para la fase de documentación, se procederá con el registro de la totalidad de los endpoints disponibles en el BackEnd. Para este fin se optó por el uso de la herramienta Postman, la cual permite la implementación de un ambiente colaborativo para el registro de estos, así como su separación en diferentes carpetas, las cuales han sido definidas en base al propósito de cada endpoint y su principal esquema objetivo. Para el sistema PREC, se definieron los siguientes grupos o colecciones de endpoints en Postman:

1. Auth: En este grupo se encuentra la documentación de la totalidad de endpoints involucrados en el proceso de inicio y mantenimiento de sesión, así como la obtención del token requerido para autorizar el acceso a varios de los endpoints disponibles en los siguientes grupos.

2. Users: La colección Users incluye el registro de los endpoints que realizan operaciones sobre los usuarios del sistema, como consulta, actualización, inserción y eliminación. Incluye además endpoints para la validación de datos del usuario como direcciones de correo electrónico y preguntas de seguridad.

3. Zones: Abarca los endpoints de consulta, actualización, inserción y eliminación de zonas, así como la consulta de las zonas que cuentan con datos de consumos registrados.

4. Buildings: Incluye los endpoints empleados en la consulta de los edificios, los cuales pertenecen a las zonas. Se incluye los endpoints para actualización, inserción y eliminación de los edificios.

5. Assets: Aglomera los endpoints de operaciones básicas (consulta, inserción, actualización, eliminación) sobre la entidad Assets.

6. Consumptions: Constituye una de las colecciones de mayor importancia, al incluir los endpoints responsables de la inserción, actualización y eliminación de consumos. Dentro de las consultas registradas en esta colección se hayan las de consumos generales, consumos por zonas, por edificios, por sensor, por rango de fechas, mes, días, e incluso la consulta de consumos por hora en un intervalo definido.

7. Recommendations: Aglomera los endpoints de operaciones básicas (consulta, inserción, actualización, eliminación) sobre la entidad Recommendations.

8. Sensors: Colección que abarca los endpoints de sensores, incluyendo su registro y consulta.

9. Events: Incluye los endpoints para la consulta, registro, actualización y eliminación de eventos.

10. Notifications: Incluye los endpoints para el registro, actualización y eliminación de las notificaciones. Incluye también enpoints para la consulta de notificaciones por alertas, acciones ejecutadas, evaluadas y eventos.

11. Files: La colección abarca los endpoints para consulta de los archivos cargados, así como endpoints para la carga y descarga de archivos.

12. Bills: Abarca los endpoints de registro de facturas, así como la consulta de facturas por medidor, descarga de facturas y descarga de reportes de facturas.

2.2.2 Monitoreo e Identificación

La fase de Monitoreo e Identificación hace referencia al proceso efectuado para la determinación de cuales endpoints presentan un malfuncionamiento. Es decir, esta fase del proceso metodológico tiene como principal objetivo la obtención de un listado de aquellos métodos disponibles en el BackEnd que presentan altos tiempos de carga. Para efectos de este proyecto, se considerarán como endpoints con altos tiempos de carga a aquellos con un tiempo promedio $t >= 5s$, tiempo en cual se estima el usuario comúnmente desiste de la aplicación como foco de atención y procede a salir de esta o intentar refrescar la página [7].

Con el fin de medir los tiempos de carga o respuesta de los endpoints documentados en la fase 1, se hará uso nuevamente de la herramienta Postman, puesto que esta permite efectuar las invocaciones de estos servicios de forma sencilla, definiendo el cuerpo de la solicitud y proveyendo de una vista para la respuesta del servidor, así como información adicional en la que se incluye el tiempo de respuesta de la petición efectuada.

Se procederá a realizar un total de treinta (30) invocaciones a cada uno de los servicios documentados, a fin de obtener estadísticas de tiempos de respuesta promedio en función de un tamaño de muestra significativo. Esto, con la finalidad de obtener métricas básicas con las cuales establecer una comparación una vez finalizado el proceso metodológico

De los resultados obtenidos de la invocación de estos servicios se identificarán aquellos que presenten altos tiempos de respuesta, los cuales serán analizados como indica la siguiente fase.

2.2.3 Análisis de Endpoints

Para el análisis de los endpoints marcados en la fase de Monitoreo e Identificación se procederá a la revisión del código fuente de los algoritmos detrás de cada endpoint, a fin de determinar su cota asintótica de complejidad haciendo uso de la notación Big O, la cual permite identificar la eficiencia de un algoritmo en función del tamaño de sus entradas, asumiendo siempre el peor escenario [8]. En función de este análisis será posible determinar si el algoritmo implementado es particularmente ineficiente y por consiguiente, si es posible su reingeniería.

2.2.4 Reingeniería

Para la fase de reingeniería, se adoptarán una o más de las siguientes estrategias para la optimización de las consultas, evidenciadas en la reducción de los tiempos de respuesta en las consultas:

1. Migración de carga de procesamiento al motor de base de datos: Reestructuración de los queries efectuados a fin de que estos realicen una mayor porción del indexado, procesamiento y cálculos, en lugar de que estos sean realizados por el servidor web.
2. Reducción de la complejidad de los algoritmos encargados del procesamiento de las respuestas de los queries efectuados.
3. Reducción del número de consultas efectuadas.
4. Desvío de consultas a base de datos, obviando el paso por el middleware de mapeo objeto-relacional

2.3 Cronograma de Actividades

A continuación, la propuesta de actividades a realizar y sus fechas de inicio y fin con miras a alcanzar los objetivos propuestos, satisfaciendo los requerimientos del cliente.

Tabla 2.1: Cronograma de actividades a realizar. [Autoría Propia]

Actividades	Fecha Inicio	Fecha Fin
Levantamiento de ambientes de desarrollo	6/3/2022	6/10/2022
Documentación de endpoints (Postman)	6/3/2022	6/17/2022
Monitoreo Inicial de Endpoints	6/3/2022	6/17/2022
Identificación de endpoints con alta latencia	6/17/2022	6/24/2022
Análisis de uso de endpoints en el FrontEnd	6/17/2022	6/24/2022
Análisis de complejidad de endpoints	6/24/2022	7/21/2022
Reingeniería de endpoints	6/24/2022	7/21/2022
Migración de modulo Muestras	7/7/2022	7/14/2022
Migración de modulo Sensores en tiempo real	7/7/2022	7/14/2022
Presentación de mejoras efectuadas a usuarios de sostenibilidad	7/7/2022	7/14/2022
Recepción de solicitudes de mejora	7/7/2022	7/14/2022
Definición de solicitudes a implementar en conjunto con el cliente	7/7/2022	7/14/2022
Monitoreo y documentación Final de Endpoints	7/14/2022	7/21/2022
Implementación de mejoras solicitadas	7/14/2022	8/12/202

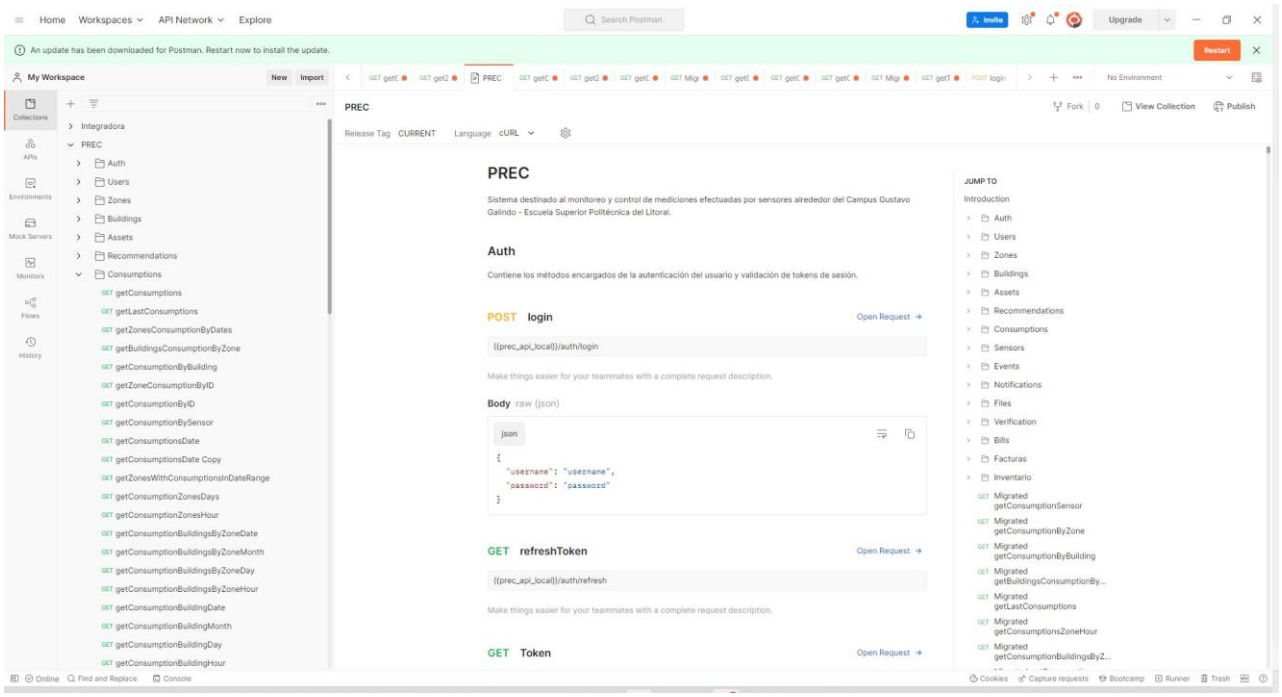
CAPÍTULO 3

3. RESULTADOS

3.1 Documentación

El proceso de documentación supuso un paso crucial en el inicio del proceso iterativo de obtención de la solución, ya que permitió la obtención de una visión clara respecto al funcionamiento del sistema en sus unidades básicas constituyentes, los endpoints. Mediante la recopilación y documentación de estos fue posible identificar, a grosso modo, las acciones que el sistema es capaz de desempeñar. Tras el proceso de documentación, se contabilizaron un total de 108 endpoints repartidos en las colecciones mencionadas durante el capítulo 2.

Figura 3.1: Vista general, Documentación de endpoints en Postman - PREC. [Autoría Propia]



Una vez documentados los endpoints, fue posible iniciar la segunda fase del proceso metodológico, la cual es Monitoreo e Identificación de Endpoints.

3.2 Monitoreo e Identificación

En la fase de monitoreo e identificación, se centraron esfuerzos en los endpoints involucrados en la recopilación de estadísticas de consumos eléctricos. Esto, acorde a lo requerido por el cliente, puesto que los usuarios reportan altos tiempos de carga en el uso de estos endpoints. Para el monitoreo de endpoints se realizo un total de 30 invocaciones a los servicios, a fin de contar con un tamaño muestral significativo. Tras este proceso de monitoreo, se tienen los siguientes resultados promedio para los endpoints existentes en el sistema y que corresponden a la obtención de estadísticas de consumo eléctrico.

Tabla 3.1: Tabla de tiempos promedios de respuesta iniciales. [Autoría Propia]

Detalle Tiempos de Respuesta	
Endpoint	Tiempo de respuesta promedio ($n = 30$)(s)
/consumptions	0.9653
/getLastConsumptions	0.0115
/consumptionZones	32.2653
/consumptionBuildingsByZone	0.8726
/consumptionBuilding	0.7073
/consumptionZone	30.8920
/consumptionSensor	0.0102
/consumptionsDate	0.3755
/cosumptionZonesMonth	0.6171
/consumptionZonesDay	0.6287
/consumptionZonesHour	58.1783
/consumptionBuildingsByZoneDate	22.1177
/consumptionBuildingsByZoneMonth	0.3528
/consumptionBuildingsByZoneDay	0.3688
/consumptionBuildingsByZoneHour	16.6887
/consumptionBuildingDate	0.0376

Como se puede evidenciar en la tabla 3.1, los resultados recopilados para los diferentes endpoints son variados, en un rango que tiene como límite inferior $11ms$ de tiempo de respuesta y $58.17s$ como límite superior. De este listado, y según el criterio establecido en el capítulo 2, es posible determinar que aquellos endpoints deben pasar a la segunda fase de análisis de complejidad asintótica son los correspondientes a:

- /consumptionZones - $32.2653s$
- /consumptionZone - $30.8920s$
- /consumptionZonesHour - $58.1783s$
- /consumptionBuildingsByZoneDate - $22.1177s$
- /consumptionBuildingsByZoneHour - $16.6887s$

En la fase de análisis de complejidad asintótica, se priorizaron los endpoints descritos anteriormente. Adicionalmente, se definió bajo sugerencia al cliente, la revisión de la totalidad de los endpoints correspondientes a la obtención de estadísticas de consumo eléctrico, puesto que, se consideró estos endpoints podrían presentar complejidades elevadas a pesar de obtener bajos tiempos de respuesta, debido a que no todos los sensores se encuentran activos, por lo que no necesariamente cuentan con volúmenes de datos similares en las consultas efectuadas. Sin embargo, cuando todos los sensores se encuentren operativos, el volumen de información para estos estará en constante crecimiento, lo que podría llevar a problemas de rendimiento en el futuro. Por esta razón, se sometió la totalidad de endpoints al análisis, a fin de identificar de forma temprana una posible falencia en el diseño de los mismos.

3.3 Analisis de Endpoints - Complejidad Asintótica

Posterior al monitoreo de Endpoints, se procedió a la determinación del orden de complejidad de los algoritmos que rigen el comportamiento de los endpoints, de forma análoga a la presentada por P.Dazinger en su publicación ”Big O Notation”. En este análisis se tomó en consideración un escenario pesimista, asumiendo un tamaño n para todas las estructuras de datos. Tras efectuado este proceso se tienen los resultados

presentados a continuación.

Para los endpoints prioritarios, identificados en la fase de monitoreo, se han determinado los siguientes órdenes de complejidad.

Tabla 3.2: Tabla de Complejidades Asintóticas - Endpoints prioritarios. [Autoría Propia]

Endpoint	Orden de Complejidad
/consumptionZones	$O(n^4)$
/consumptionZone	$O(n^3)$
/consumptionZonesHour	$O(n^3)$
/consumptionBuildingsByZoneDate	$O(n^2)$
/consumptionBuildingsByZoneHour	$O(n^3)$

Como se evidencia en la tabla 3.2, el endpoint que obtuvo un mayor orden de complejidad es ”/consumptionZones” con una complejidad de orden n^4. Luego, con un orden de complejidad n^3 siguen los endpoins ”/consumptionZone”, ”/consumptionZonesHour” y ”/consumptionBuildingsByZoneHour”. Finalmente, de estos endpoints, el de menor complejidad es ”/consumptionBuildingsByZoneDate” con orden n^2.

De los endpoints restantes se obtuvieron los siguientes órdenes de complejidad:

De forma similar a los resultados obtenidos para los endpoints prioritarios, podemos encontrar órdenes de complejidad n^4, n^3 y n^2. Como particularidad, entre los endpoints no prioritarios encontramos órdenes de complejidad adicionales, como n y 1, dónde 1 denota una complejidad no dependiente del tamaño de una colección o estructura de datos, es decir, constante.

Con esta información, es posible la elección de la estrategia a adoptar para la migración de cada endpoint, de entre aquellas definidas en la sección de Metodología, subsección 2.2.4.

3.4 Reingeniería de Endpoints

En esta sección se describirán las estrategias adoptadas como parte de la reingeniería de los diferentes componentes que conforman el sistema PREC, incluyendo los endpoints en el BackEnd así como las interfaces de usuario y la lógica detrás de estas en el FrontEnd.

Tabla 3.3: Tabla de Complejidades Asintóticas - Endpoints no prioritarios. [Autoría Propia]

Endpoint	Orden de Complejidad
/consumptions	$O(1)$
/lastConsumptions	$O(n)$
/consumptionBuildingsByZone	$O(n^3)$
/consumptionBuilding	$O(n^2)$
/consumptionSensor	$O(n)$
/consumptionsDate	$O(1)$
/consumptionZonesMonth	$O(n^4)$
/consumptionZonesDay	$O(n^3)$
/consumptionBuildingsByZoneMonth	$O(n^3)$
/consumptionBuildingsByZoneDay	$O(n^3)$
/consumptionBuildingDate	$O(1)$

3.4.1 BackEnd

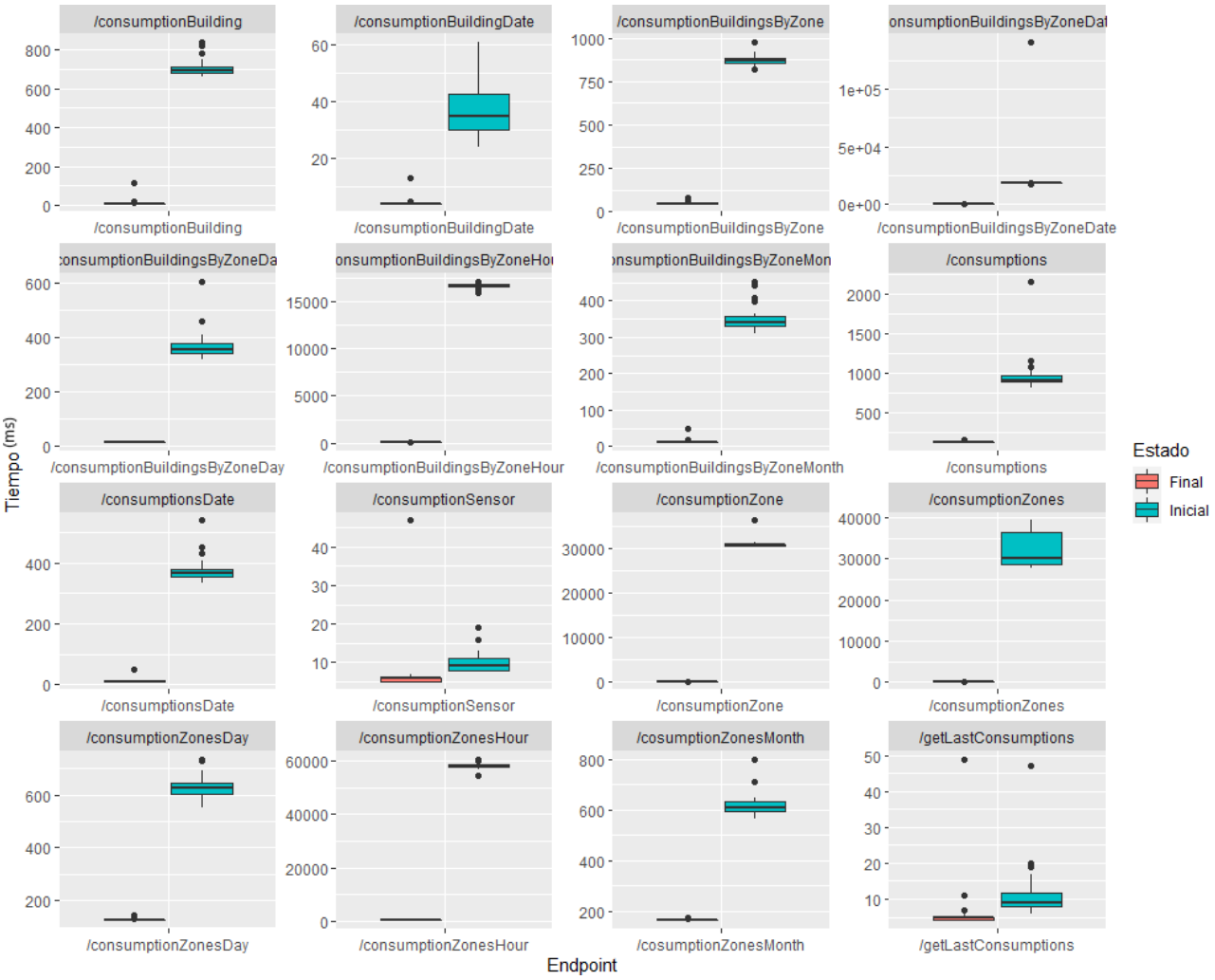
A continuación, el detalle de estrategias de optimización adoptadas, así como los resultados obtenidos tras el proceso de reingeniería para cada uno de los endpoints.

Tabla 3.4: Tabla de resultados post proceso de Reingeniería. [Autoría Propia]

Endpoint	Estrategias	Complejidad Inicial	Complejidad Final	Tiempo Inicial(s)	Tiempo Final(s)
/consumptions	4	$O(1)$	$O(1)$	0.9653	0.1210
/getLastConsumptions	1, 2, 4	$O(n)$	$O(1)$	0.0115	0.0065
/consumptionZones	1, 2, 4	$O(n^4)$	$O(n)$	32.2653	0.1284
/consumptionBuildingsByZone	1, 2, 4	$O(n^3)$	$O(n)$	0.8726	0.0508
/consumptionBuilding	1, 2, 4	$O(n^2)$	$O(n)$	0.7073	0.0117
/consumptionZone	1, 2, 4	$O(n^3)$	$O(n)$	30.8920	0.0425
/consumptionSensor	4	$O(n)$	$O(n)$	0.0102	0.0070
/consumptionsDate	4	$O(1)$	$O(1)$	0.3755	0.0115
/cosumptionZonesMonth	1, 2, 4	$O(n^4)$	$O(n)$	0.6171	0.1665
/consumptionZonesDay	1, 2, 4	$O(n^4)$	$O(n)$	0.6287	0.1279
/consumptionZonesHour	1, 2, 4	$O(n^3)$	$O(n)$	58.1783	0.1303
/consumptionBuildingsByZoneDate	1, 2, 4	$O(n^2)$	$O(n)$	22.1177	0.0129
/consumptionBuildingsByZoneMonth	1, 2, 4	$O(n^3)$	$O(n)$	0.3528	0.0131
/consumptionBuildingsByZoneDay	1, 2, 4	$O(n^3)$	$O(n)$	0.3688	0.0119
/consumptionBuildingsByZoneHour	1, 2, 4	$O(n^3)$	$O(n)$	16.6887	0.0263
/consumptionBuildingDate	4	$O(1)$	$O(1)$	0.0376	0.0044

Como se puede evidenciar en la tabla 3.4, existen cambios en los tiempos promedios de consulta para la totalidad de los endpoints sometidos al proceso de reingeniería. Nótese, como aquellos endpoints clasificados como prioritarios en la fase de monitoreo e identificación presentan una reducción de hasta $58.04s$, la cual representa aproximadamente el 99.77% del tiempo de carga original. Esto, tras el proceso de reingeniería de los mismos, en el que a todos estos endpoints se aplicaron las estrategias 1, 2 y 4 en conjunto, que consisten en la reducción del orden de complejidad, redirección de las consultas obviando el paso por el middleware de mapeo objeto-relacional y la migración de las consultas efectuadas a fin de descargar parte de los calculos sobre el motor de base de datos y no el servidor web.

Figura 3.2: Endpoints migrados - PREC. [Autoría Propia]



El impacto del proceso de reingeniería de endpoints puede ser evidenciado también

de forma gráfica, como muestra la figura 3.2 Para cada endpoint migrado se puede encontrar una subgráfica, la cual muestra los diagramas de caja y bigote para el endpoint antes y después de ser sometido a la reingeniería. Nótese como en todos los casos, los diagramas del costado izquierdo, correspondientes a los tiempo de respuesta post migración se encuentran más abajo en la recta numérica, lo que denota menores tiempos. Nótese también que las cajas son, por lo general, mas compactas, con una menor distancia entre sus extremos, lo que denota a su vez una mayor consistencia en los tiempos de respuesta, es decir, estos poseen una varianza menor.

Cada uno de los escenarios mostrados a través de la figura 3.2 pueden ser visualizados, a mayor detalle, en el Apéndice A, donde se presenta una figura por cada endpoint.

Nótese además la reducción del orden de complejidad en algunos de los endpoints, como /consumptionZones, /consumptionZonesMonth o /consumptionZonesDay, los cuales presentaban una complejidad $O(n^4)$ y tras la migración de los mismos, todos presentan una complejidad $O(n)$ lo cual quiere decir que el tiempo de carga del endpoint crece en una relación lineal con el tamaño de los datos, en comparación al crecimiento con potencia a la cuarta que mantenían. Algo similar ocurre con la mayoría de los endpoints restantes, en los que se ha podido reducir también sus órdenes de complejidad.

Existen también, sin embargo, endpoints para los cuales no era factible una reducción del órden de complejidad, ya sea porque esta era constante, o porque esta era necesaria para realizar el procesamiento deseado sobre los datos. Este es el caso de endpoints como /consumptionZone, en el cual se mantuvo una complejidad $O(n)$.

Una vez concluído el proceso de reingeniería y posterior monitoreo de los endpoints migrados, a fin de conocer sus nuevos tiempos de respuesta promedio, se procedió con la migración de los métodos que invocaban a estos desde el FrontEnd, es decir, desde la aplicación con interfaz gráfica con la que interactúa el usuario.

3.4.2 FrontEnd

En el FrontEnd se efectuaron dos clases de trabajos distintos. La primera, abarca los trabajos de migración consecuentes a la reingeniería de los endpoints, la cual tuvo lugar en el BackEnd. Estos trabajos de migración serán cubiertos en la sección Migración de consultas. La segunda clase de trabajos de migración es aquella realizada sobre los módulos del sistema, bajo solicitud del cliente y los usuarios finales del sistema. Estos

cambios serán tratados de forma independiente en una sección para cada módulo.

3.4.3 Migración de consultas

Para la migración de consultas, se procedió a la identificación de las líneas de código que incluían el nombre de cada uno de los endpoints migrados. Esto por medio de la herramienta de búsqueda que incorpora Visual Studio Code. Esta herramienta presenta un listado de todas las ocurrencias, en este caso, llamadas al endpoint.
En el caso del proyecto PREC, las llamadas a los endpoints se encuentran encapsuladas por medio de servicios de Angular, los cuales suponen una abstracción de la invocación del protocolo HTTP, permitiendo al desarrollador realizar la consulta al endpoint deseado como una función.
Se procedió con la migración del endpoint al que apuntan estos servicios, reemplazando la url del endpoint original por el endpoint migrado. Esto es posible debido a que los endpoints migrados mantienen consistencia en lo que a la respuesta respecta. Es decir, las respuestas de los endpoints migrados mantienen el mismo formato que las respuestas de los endpoints originales, a fin de facilitar su migración desde el FrontEnd.

Figura 3.3: Endpoints migrados - PREC. [Autoría Propia]

Como se muestra en la figura 3.3, la migración de las consultas es realizada a nivel del servicio de Consulta de Consumos, que contiene los diferentes microservicios para llamada a los endpoints.

Figura 3.4: Ejemplo de microservicio - PREC. [Autoría Propia]

```
getLastConsumption(): Observable<any> {
  const token = localStorage.getItem('tokenUser');
  const options = { headers: new HttpHeaders({ 'Content-Type': 'application/json', 'access-token': token }) };
  const endpoint = 'migrated/lastConsumptions';
  return  this._http.get<any>(this._apiUrl + endpoint, options)
      .do(
          data => { },
          (err: HttpErrorResponse) => { }
      ).catch(this.handleError);
}
```

La figura 3.4 muestra un ejemplo de microservicio dentro del servicio de Consulta de Consumos. En este caso, se trata del microservicio getLastConsumption el cual invoca al endpoint /lastConsumptions migrado.

Figura 3.5: Ejemplo de invocación a microservicio - PREC. [Autoría Propia]

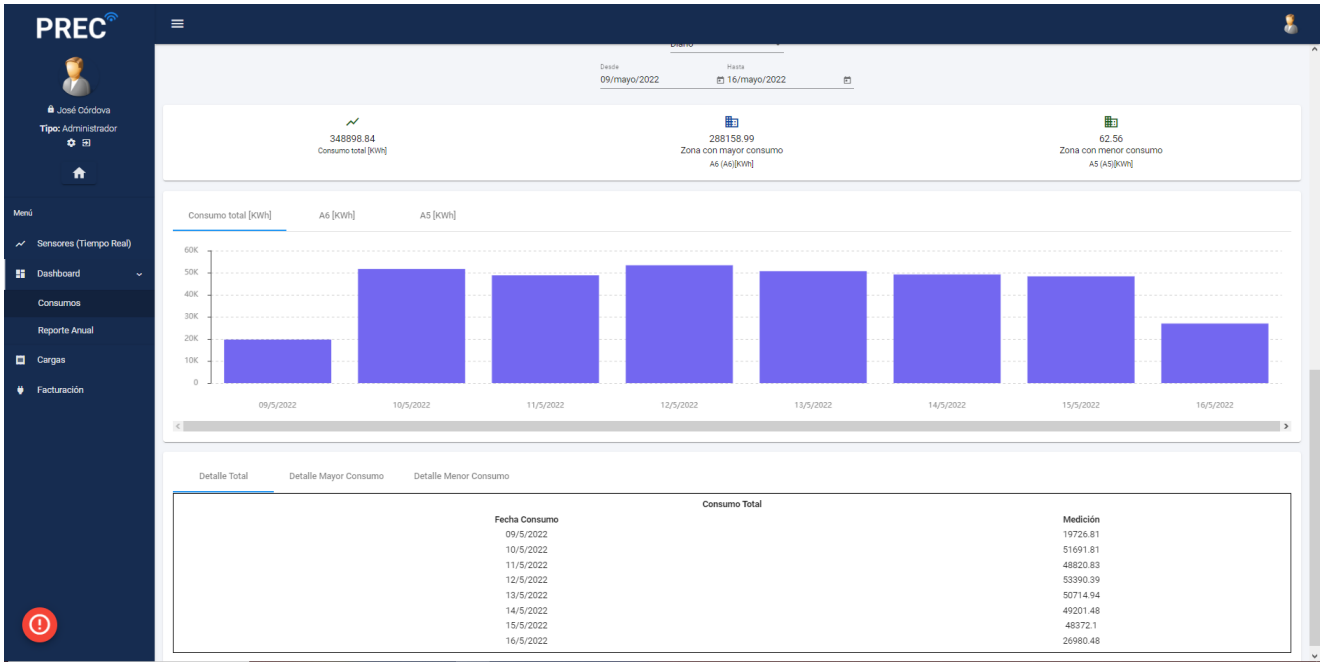
```
/**
 * Print the data to the chart
 * @function showData
 * @return {void}
 */
private showData(): void {
  //Dynamic graphic
  this.consumptionsService.getLastConsumption().subscribe(response => {···
  });
}
```

La figura 3.5 ejemplifica como son invocados los microservicios migrados. Nótese como estos son llamados como si fueran funciones, abstrayendo al programador del uso del protocolo HTTP. La función retorna un objeto de tipo Observable, al cual se ha de suscribir, a fin de recibir la información contenida en la respuesta del endpoint.

3.4.4 Módulo Muestras

Dentro del módulo muestras, los cambios implementados corresponden a la adición de un detalle de los consumos consultados para las zonas, solicitado puesto que los usuarios desean poder consultar la información en forma de tabla adicional al gráfico mostrado.

Figura 3.6: Tabla de consumos registrados - PREC. [Autoría Propia]

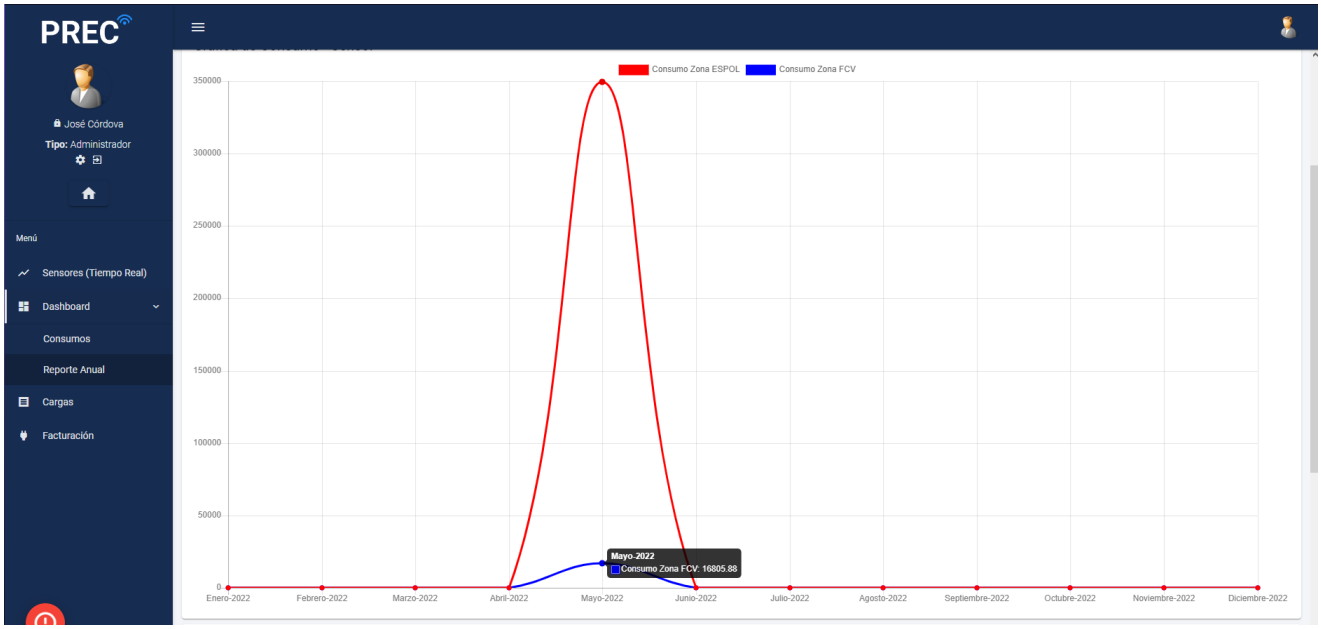


La figura 3.6 muestra el módulo de muestras, ahora renombrado a módulo de consumos bajo petición de los usuarios del sistema. Nótese como bajo la gráfica de consumos aparece una tabla que detalla los valores registrados en el gráfico. Esta tabla se muestra para los consumos de las zonas, incluyendo general, zona de mayor consumo y zona de menor consumo.

3.4.5 Módulo Reporte Anual

El módulo de reporte anual es un módulo completamente nuevo dentro del Sistema PREC, introducido como parte de los desarrollos efectuados bajo solicitud de los usuarios del sistema. En este módulo, es posible la comparativa entre dos zonas a elección del usuario, en un intervarlo definido por el mismo. Esta comparativa es efectuada por años, mostrando un detalle mensual. Esta información recopilada es empleada para la generación de un gráfico como se muestra a continuación.

Figura 3.7: Gráfica comparativa, Reporte Anual - PREC. [Autoría Propia]



La figura 3.7 muestra la gráfica que se genera, al seleccionar la zona ”ESPOL”, que engloba a todas las zonas y la zona “FCV”.

Adicional a la gráfica, se genera un reporte en forma de tabla, que muestra la misma información que el gráfico.

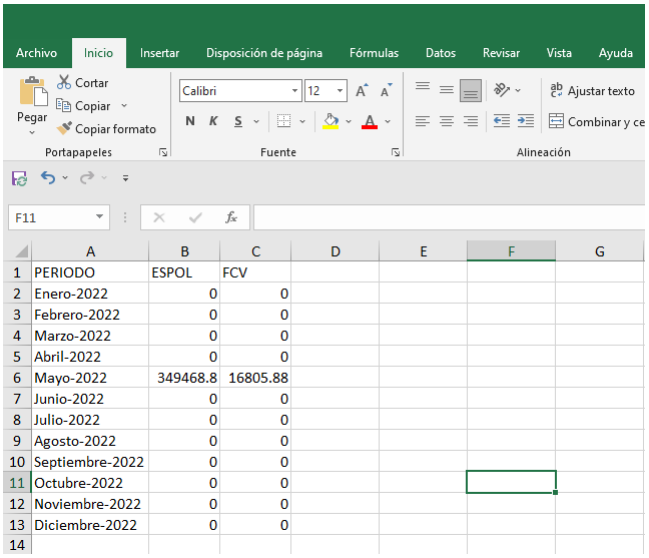
Figura 3.8: Detalle en forma de Tabla, Reporte Anual - PREC. [Autoría Propia]

Descargar Reporte

Periodo	Consumo ESPOL [KWh]	Consumo FCV [KWh]
Enero-2022	0	0
Febrero-2022	0	0
Marzo-2022	0	0
Abril-2022	0	0
Mayo-2022	349468.8	16805.88
Junio-2022	0	0
Julio-2022	0	0
Agosto-2022	0	0
Septiembre-2022	0	0
Octubre-2022	0	0
Noviembre-2022	0	0
Diciembre-2022	0	0

La figura 3.8 muestra la tabla generada, la misma que es posible descargar en formato excel (.xlsx) para su uso en reportería, como muestra la figura 3.8.

Figura 3.9: Reporte en formato Excel, Reporte Anual - PREC. [Autoría Propia]



	A	B	C	D	E	F	G
1	PERIODO	ESPOL	FCV				
2	Enero-2022	0	0				
3	Febrero-2022	0	0				
4	Marzo-2022	0	0				
5	Abril-2022	0	0				
6	Mayo-2022	349468.8	16805.88				
7	Junio-2022	0	0				
8	Julio-2022	0	0				
9	Agosto-2022	0	0				
10	Septiembre-2022	0	0				
11	Octubre-2022	0	0				
12	Noviembre-2022	0	0				
13	Diciembre-2022	0	0				
14							

3.4.6 Módulo Sensores en Tiempo Real

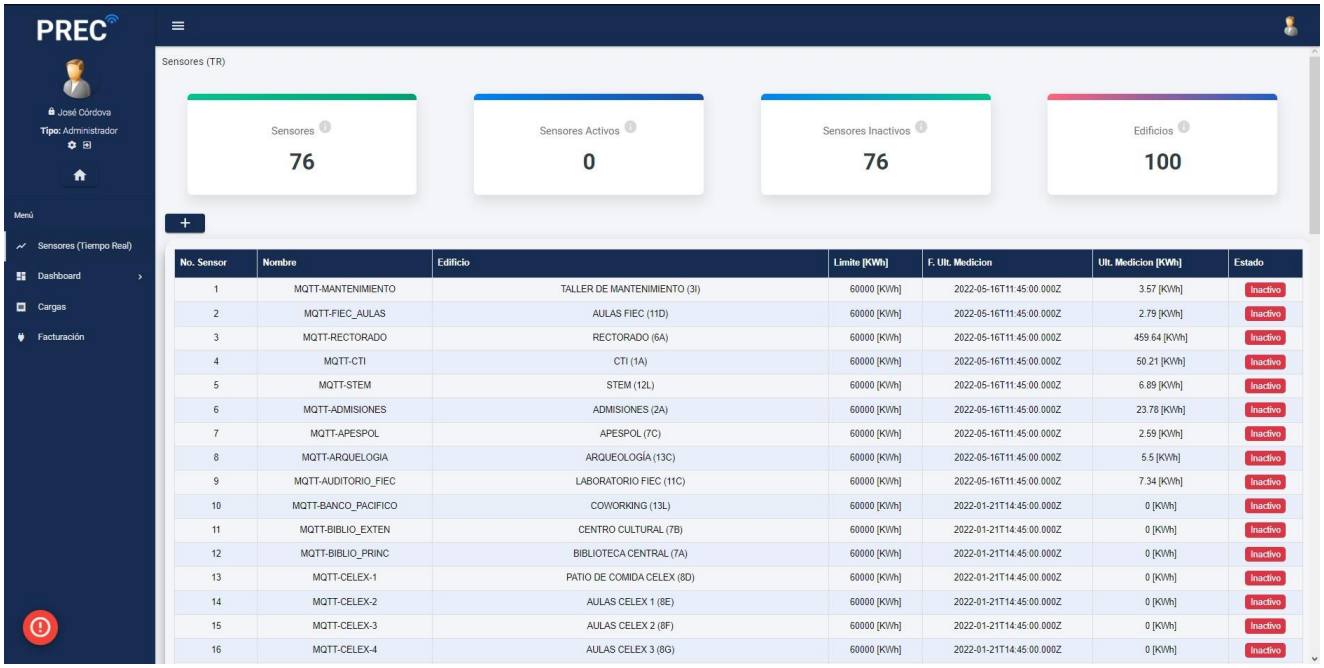
En el módulo de sensores en tiempo real se han introducido varios cambios, que afectan tanto su estética como su funcionamiento.

Para comenzar, se ha reemplazada la vista de cartillas que este módulo mantenía para cada sensor, reemplazandola por una vista de tabla, que permite mostrar más información haciendo uso de menos espacio.

En segundo lugar, se ha reemplazado la renderización de las gráficas consumo para cada sensor en tiempo de carga por la renderización de gráficas individuales bajo demanda. Esto quiere decir, las gráficas ya no son cargadas todas una vez que el usuario ingresa al módulo, puesto que esto ocasionaba grandes tiempos de carga. En su lugar, la gráfica es generada bajo demanda una vez el usuario selecciona un sensor, lo que permite presentar información relevante y actualizada al usuario en el momento en que este la desee consultar, en lugar de mostrar toda la información disponible desde un inicio, sin tener en consideración si es lo que el usuario desea ver.

Adicionalmente, se incluyó una sección de datos generales, en forma de cartillas, que muestran breves resumenes de información como el número total de sensores, el conteo de sensores activos, inactivos y el número de edificios registrados en el sistema.

Figura 3.10: Tabla de Sensores - PREC. [Autoría Propia]



No. Sensor	Nombre	Edificio	Limite [KWh]	F. Ult. Medicion	Ult. Medicion [KWh]	Estado
1	MQTT-MANTENIMIENTO	TALLER DE MANTENIMIENTO (3I)	60000 [KWh]	2022-05-16T11:45:00.000Z	3.57 [KWh]	Inactivo
2	MQTT-FIEC_AULAS	AULAS FIEC (11D)	60000 [KWh]	2022-05-16T11:45:00.000Z	2.79 [KWh]	Inactivo
3	MQTT-RECTORADO	RECTORADO (6A)	60000 [KWh]	2022-05-16T11:45:00.000Z	459.64 [KWh]	Inactivo
4	MQTT-CTI	CTI (1A)	60000 [KWh]	2022-05-16T11:45:00.000Z	50.21 [KWh]	Inactivo
5	MQTT-STEM	STEM (12L)	60000 [KWh]	2022-05-16T11:45:00.000Z	6.89 [KWh]	Inactivo
6	MQTT-ADMISIONES	ADMISIONES (2A)	60000 [KWh]	2022-05-16T11:45:00.000Z	23.78 [KWh]	Inactivo
7	MQTT-APESPOL	APESPOL (7C)	60000 [KWh]	2022-05-16T11:45:00.000Z	2.59 [KWh]	Inactivo
8	MQTT-ARQUELOGIA	ARQUEOLOGÍA (13C)	60000 [KWh]	2022-05-16T11:45:00.000Z	5.5 [KWh]	Inactivo
9	MQTT-AUDITORIO_FIEC	LABORATORIO FIEC (11C)	60000 [KWh]	2022-05-16T11:45:00.000Z	7.34 [KWh]	Inactivo
10	MQTT-BANCO_PACIFICO	COWORKING (13L)	60000 [KWh]	2022-01-21T14:45:00.000Z	0 [KWh]	Inactivo
11	MQTT-BIBLIO_EXTEN	CENTRO CULTURAL (7B)	60000 [KWh]	2022-01-21T14:45:00.000Z	0 [KWh]	Inactivo
12	MQTT-BIBLIO_PRINC	BIBLIOTECA CENTRAL (7A)	60000 [KWh]	2022-01-21T14:45:00.000Z	0 [KWh]	Inactivo
13	MQTT-CELEX-1	PATIO DE COMIDA CELEX (8D)	60000 [KWh]	2022-01-21T14:45:00.000Z	0 [KWh]	Inactivo
14	MQTT-CELEX-2	AULAS CELEX 1 (8E)	60000 [KWh]	2022-01-21T14:45:00.000Z	0 [KWh]	Inactivo
15	MQTT-CELEX-3	AULAS CELEX 2 (8F)	60000 [KWh]	2022-01-21T14:45:00.000Z	0 [KWh]	Inactivo
16	MQTT-CELEX-4	AULAS CELEX 3 (8G)	60000 [KWh]	2022-01-21T14:45:00.000Z	0 [KWh]	Inactivo

La figura 3.10 muestra la tabla de sensores, incluyendo información relevante respecto al mismo, como su nombre, ubicación, estado, etc.

Figura 3.11: Gráfica de consumos de sensor bajo demanda - PREC. [Autoría

La figura 3.11 muestra la gráfica de los ultimos consumos efectuados por el sensor. La gráfica es generada bajo demanda al hacer clic en uno de los sensores de la tabla mostrada en la figura 3.9.

3.5 Análisis de Costos

Una vez concluído el proceso de desarrollo de las soluciones propuestas en el capítulo 2, se procedió con el análisis de costos, el cual sirve de ejemplo, ilustrando el costo que la implementación de la solución hubiera representado para el cliente. En este caso en particular, se presentará el detalle de actividades realizadas, así como el tiempo estimado invertido en estas. El costo de estas actividades fué calculado tomando como base un salario de $1000 a tiempo completo, es decir, 40 horas semanales considerando un mes de 4 semanas. Esto establece un costo por hora de $6.25 el mismo que fue empleado para el cálculo de los totales.

Tabla 3.5: Tabla de resultados post reingeniería. [Autoría Propia]

Actividades	Horas	Costo ($)
Levantamiento ambientes de desarrollo	5	31.25
Documentación Endpoints	5	31.25
Monitoreo Inicial e Identificación de Endpoints	10	62.5
Análisis de Endpoints	20	125
Reingeniería Endpoints	50	312.5
Migración módulo Muestras	20	125
Migración módulo Sensores (TR)	20	125
Elicitación de requerimientos (usuarios)	10	62.5
Implementación de mejoras convenidas	20	125
Monitoreo y documentación final	20	125
Despliegue en ambiente pre - productivo	5	31.25
Total	185	1156.25

La tabla 3.5 muestra los totales estimados por actividad realizada, así como el total final del proyecto. El costo final asciende a $1156.25.

3.6 Cierre de Proyecto

Para la culminación de este proyecto, tuvo lugar una reunión con el cliente, en la cual se iteró sobre los diferentes módulos del sistema, evidenciando la implementación de los cambios convenidos. Posterior a esto, se procedió a compartir con el cliente un acta de aceptación, la cual detalla los entregables finales. En esta acta consta la firma del cliente.

Los entregables son:

- Documentación de endpoints en PostMan.
- Código fuente BackEnd en Git.
- Código fuente FrontEnd en Git.

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

El proceso de Documentación de endpoints permitió obtener valiosa información respecto a la forma en la que estaba construido el sistema. Esto facilitó los trabajos efectuados como parte de la fase de Monitoreo e Identificación de endpoints, permitiendo realizar pruebas sobre los endpoints de forma rápida y sencilla, que a su vez permitió segregar aquellos endpoints que requerían de mayor análisis y eventualmente una reingeniería.

Las estrategias adoptadas en el proceso de reingeniería demostraron ser efectivas, puesto que se alcanzó una reducción de tiempos de carga en todos los endpoints sometidos a la reingeniería. Asimismo, la elección de las estrategias aplicada a cada endpoint quedó justificada con los resultados de tiempos de carga post reingeniería.

Los cambios realizados en el FrontEnd también demostraron ser efectivos, puesto que se redujeron los tiempos de carga en los módulos de Sensores en Tiempo Real y en el módulo de Muestras. En el caso del módulo de sensores, la reducción de tiempos de carga viene dada por el cambio en la estrategia de renderización de gráficos, descartando la carga de todos los gráficos, que eran 76, al haber 76 sensores registrados, en favor de cargar cada gráfico bajo demanda.

En el módulo de muestras se logró reducir los tiempos de carga en las consultas y generación de gráficos, puesto que estas dependían en su totalidad de los microservicios que consultaban a los endpoints sometidos al proceso de reingeniería.
A manera de resumen, el trabajo realizado como parte de este proyecto de Materia

Integradora es de gran importancia, porque supone una reducción considerable en los tiempos de carga de un sistema con un uso creciente como lo es el sistema PREC. Esta optimización del sistema supone una mejor experiencia para los usuarios del departemento de Sostenibilidad de ESPOL, para quienes la demora en el sistema representaba un punto de dolor.

Adicionalmente, las nuevas funcionalidades añadidas al sistema permitirán al usuario acceder a información de calidad, la cual es empleada por ESPOL en la elaboración de informes de sostenibilidad, los cuales son compartidos a los entes reguladores y de ranking pertinentes.

4.2 Recomendaciones

Si bien es cierto que el objetivo de este proyecto es la optimización y expansión del sistema PREC, durante la ejecución del mismo se han descubierto nuevas oportunidades de mejora para el sistema. Entre las cuales se destacan las siguientes.

- Actualización del sistema: El sistema PREC, como fue mencionado en el Capítulo 1, está construido, en lo que a FrontEnd respecta, haciendo uso del Framework Angular. Sin embargo, la versión que este maneja (8.0.0) se encuentra ya desactualizada, por lo que carece de soporte y cada vez son menos las librerías o dependencias que la incluyen como parte de las versiones que soportan. Por ende, es necesaria la migración del FrontEnd, a versiones de Angular más actuales, que garanticen la posibilidad de expansión de las funcionalidades del sistema, puesto que aceptan un mayor número de librerías o dependencias multipropósito.

- Migración de consultas a esquemas: Como parte de este proyecto integrador, se realizó el análisis y eventual reingeniería de las consultas enfocadas a obtención de estadísticas de consumos, sin embargo, existen más esquemas hacia los cuales se efectúan consultas. Es por esto, que se recomienda el análisis y reingeniería de los endpoints existentes para los demás esquemas disponibles en la base de datos.

BIBLIOGRAFÍA

[1] L. J. Aguilar, Big Data, Análisis de grandes volúmenes de datos en organizaciones. Alfaomega Grupo Editor, 2016.

[2] I. Lee and K. Lee, “The internet of things (iot): Applications, investments, and challenges for enterprises,” Business horizons, vol. 58, no. 4, pp. 431–440, 2015.

[3] A. J. Poulter, S. J. Johnston, and S. J. Cox, “Using the mean stack to implement a restful service for an internet of things application,” in 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), pp. 280–285, IEEE, 2015.

[4] S. Rautmare and D. Bhalerao, “Mysql and nosql database comparison for iot application,” in 2016 IEEE international conference on advances in computer applications (ICACA), pp. 235–238, IEEE, 2016.

[5] M. Heller, “What is node. js? the javascript runtime explained,” InfoWorld, 2017.

[6] T. Holowaychuk, “Express: Fast, unopinionated, minimalist web framework for node,” Accessed: Sep, vol. 22, p. 2020, 2020.

[7] DNSstuff, “Best server and application response time monitoring tools + guide,” Sep 2020.

[8] P. Danziger, “Big o notation,” Source internet: http://www. scs. ryerson. ca/~ mth110/Handouts/PD/bigO. pdf, Retrieve: April, 2010.

APÉNDICES

Apéndice A

Carta de Aceptación del cliente

Lunes, 5 de Septiembre del 2022

Acta de Entrega

Por medio de la presente se deja constancia de la entrega y aceptación por parte del cliente, de los siguientes entregables, después de haber sido revisados y probados en reuniones efectuadas entre el cliente y el alumno de materia integradora.

Entregable	Ubicación
Código Fuente Servidor Web BackEnd	Repositorio del cliente en Git
Código Fuente Aplicativo FrontEnd	Repositorio del cliente en Git
Documentación de Endpoints Existentes	Proyecto Postman Colaborativo

Los entregables contienen los cambios realizados como parte del proyecto integrador "Optimización y Expansión de Módulos del Sistema PREC para edificios inteligentes".

Firmado electrónicamente por:
ALBERTO JOSE
CRUZ OCHOA

Ing. Alberto José Cruz Ochoa
Cliente

Sebastian Mendoza Ramírez
Alumno