PRACTICA EMPRESARIAL: REFACTORIZACIÓN DEL SISTEMA DE INFORMACIÓN *SOST* PARA SOPORTAR UNA ACTUALIZACIÓN TECNOLÓGICA, TANTO GRÁFICA COMO FUNCIONAL, BASADO EN UN PATRÓN MVC.

LUIS CARLOS ÁLVAREZ ALVAREZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2018

PRACTICA EMPRESARIAL: REFACTORIZACIÓN DEL SISTEMA DE INFORMACIÓN *SOST* PARA SOPORTAR UNA ACTUALIZACIÓN TECNOLÓGICA TANTO GRÁFICA COMO FUNCIONAL, BASADO EN UN PATRÓN MVC

LUIS CARLOS ALVAREZ ALVAREZ

TRABAJO DE GRADO PARA OPTAR AL TÍTULO DE
INGENIERO DE SISTEMAS

DIRECTOR
FERNANDO RUIZ DIAZ
M.Sc. Ingeniero de Sistemas

TUTOR
JHON EDINSON BONILLA
M.Sc. Ingeniero de Sistemas

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2018

DEDICATORIA

Mi proyecto de grado lo quiero dedicar primero que todo a Dios, que hace todo posible y que hoy me tiene redactando este proyecto, para cumplir una de muchas metas que faltan por cumplir.

Que no se quede atrás mi madre MARLENY ALVAREZ ya que los esfuerzos de esa mujer día a día me enseñaron, que la vida requiere de sacrificio y dedicación para alcanzar los sueños propuestos, también a mi padre CARLOS ALVAREZ ya que, con su poco afecto, pero con sus enseñanzas y obras realizadas me dan la motivación se salir a delante y poder realizar todo lo que él ha hecho posible.

A mi hermano NICOLAS ALVAREZ siendo el menor me dio fortalezas para seguir adelante, a mi familia en general le quiero dedicar este logro y más los que están en el cielo, ya que manifestaron alegría al enterarse de mi ingreso a la universidad, deben estar orgulloso de este logro yo sé que ellos desde arriba me guiaron por el camino correcto.

Finalmente le dedico este logro a NATALIA SOLANO ella merece esta dedicación pues, es una mujer muy especial en mi vida, siempre estuvo conmigo en la cosa buena como en las malas apoyándome a salir adelante.

AGRADECIMIENTOS

Primero que todo agradecerle a Dios pues me guio por el camino que me tenía destinado en esta vida, sin dudar le agradezco a la Universidad Industrial de Santander por recibirme en sus instalaciones y campus ofreciéndome la educación que hoy me forman una mejor persona.

Agradezco también a mis profesores por la dedicación y enseñanzas de los cuales aprendí cosas muy buenas, a mis amigos de la universidad, colegas que siempre estuvieron ahí aportando significativamente para poder cumplir esta meta.

Por último, agradecer a Movilidad y Servicios Girón s.a.s, gracias a esta entidad pude culminar mis estudios terminando con una gran enseñanza de responsabilidad, puntualidad, compañerismo, disciplina y a vivir el ambiente laboral que me forman más como persona, donde también quiero agradecer al ing. John Bonilla por la dedicación y sabiduría que me brindo guiando este proyecto para finalizarlo de la mejor manera.

TABLA DE CONTENIDO

Pág.

INTRODUCCIÓN 15
1. MOVILIDAD Y SERVICIOS GIRÓN S.A.S. 17
1.1 MISIÓN: 17
1.2 VISIÓN: 17
2. OBJETIVOS 18
2.1 OBJETIVO GENERAL: 18
2.2 OBJETIVOS ESPECÍFICOS: 18
3. MARCO TEÓRICO 19
3.1 CONTEXTUALIZACIÓN DEL PROYECTO 19
3.1.1 Análisis 19
3.1.1.1 Análisis Estructural. 20
3.1.2 Refactorización. 25
3.1.2.1 Modelo vista controlador (m.v.c). 25
3.1.2.2 Reestructuración. 27
3.2 HERRAMIENTAS UTILIZADAS EN EL DESARROLLO DEL PROYECTO 30
3.2.1 Sost. 30
3.2.1.1 Apache. 30
3.2.1.2 Php 31
3.2.1.3 JAVASCRIPT. 31
3.2.1.4 Html 31
3.2.1.5 Css. 31
3.2.1.6 Jquery. 31
3.2.1.7 Ajax. 32
3.2.1.8 Mysql 32
3.2.2 Hardware 32
3.2.2.1 Hpe proliant ml110 hardware 32
4. MARCO METODOLÓGICO 34
4.1 TAREA 1: ANÁLISIS. 35
4.1.1 Adaptación. 35
4.1.2 Planificación de la iteración. 35
4.1.3 Ejecución de la iteración. 36
4.1.4 Inspección y adaptación 36
4.2 TAREA 2: ORGANIZACIÓN DE ARCHIVOS Y COGIDO FUENTE SEGÚN EL (M.V.C). 36
4.2.1 Adaptación 36
4.2.2 Planificación de la iteración. 37
4.2.3 Ejecución de la iteración 37

4.2.3.1 Exclusión de funciones JavaScript. 37
4.2.3.2 Adaptación del nuevo encabezado a los archivos index e index_sql. 39
4.2.3.3 Exclusión de código css. 42
4.2.3.4 Exclusión de llamado Ajax. 45
4.2.3.5 Adaptación general del nuevo archivo funsiones.js. 47
4.2.4 Inspección y adaptación 49
4.3 TAREA 3: ADAPTACIÓN DE INTERFAZ GENERAL Y PLUGIN DE *ALERT*AS (INFORMATIVAS) 49
4.3.1 Adaptación. 49
4.3.2 Planificación de la iteración 49
4.3.3 Ejecución de la iteración. 49
4.3.3.1 Implementación de nueva interfaz gráfica general. 49
4.3.3.2 Implementación de plugin en *Alert*as informativas. 60
4.3.3.3 Implementación De *Loader*. 69
4.3.4 Inspección y adaptación. 70
4.4 TAREA 4: ADAPTACIÓN DE CLASES EN LOS LLAMADOS *MySQL*. 71
4.4.1 Adaptación. 71
4.4.2 Planificación de la iteración. 71
4.4.3 Ejecución de la iteración. 71
4.4.4 Inspección y adaptación. 76
4.5 TAREA 5: PRUEBAS DE FUNCIONAMIENTO DE CADA MODULO SIN IMPLEMENTAR LAS NUEVAS VERSIONES. 76
4.5.1 Adaptación. 76
4.5.2 Planificación de la iteración. 76
4.5.3 Ejecución de la iteración. 77
4.5.4 Inspección y adaptación. 78
4.6 TAREA 6: PRUEBAS DE RENDIMIENTO IMPLEMENTANDO LAS NUEVAS VERSIONES. 79
4.6.1 Adaptación. Se 79
4.6.2 Planificación de la iteración. 79
4.6.3 Ejecución de la iteración. 80
4.6.3.1 Prueba De Rendimiento Php. 83
4.6.3.1 Prueba de Velocidad Web. 85
5. CONCLUSIONES 94
6. RECOMENDACIONES 95
BIBLIOGRAFÍA 96
ANEXOS 98

TABLA DE FIGURAS

Figura 1. Diagrama Clases (Uml) - Proceso Comparendos.....21
Figura 2. Diagrama Clases (Uml) - Proceso Juridica.....21
Figura 3. Diagrama Clases (Uml) - Proceso Liquidaciones. 22
Figura 4. Diagrama Clases (Uml) - Proceso Cargues Y Envios.....23
Figura 5. Diagrama Clases (Uml) - Proceso Tramites. 24
Figura 6. Implementación De La Reestructuración.....27
Figura 7. Resultados De *Script* Version Php 5.3.8.....84
Figura 8. Resultados De *Script* Version Php 7.2.5.....84

LISTA DE ILUSTRACIONES

Ilustración 1. Instalaciones Movilidad Y Servicios Giron S.A.S.....16
Ilustración 2. Archivos (Modulo Reporte Data Crédito).....26
Ilustración 3. Archivos (Modulo Archivos Para Archivar). 26
Ilustración 4. Código Fuente Index.Php.....28
Ilustración 5. Código Fuente Index_Sql.Php.....29
Ilustración 6. Hpe Proliant Ml110.....33
Ilustración 7. Metodología Scrum 35
Ilustración 8. Reestructuración Index.Php 38
Ilustración 9. Nuevo Archivo Javascript(Funciones.Js).....39
Ilustración 10. Encabezado Simplificado De Cada Modulo Index.Php 40
Ilustración 11. Comparación Reestructuración Index_Sql.Php.....41
Ilustración 12. Bitácora.Php.....42
Ilustración 13. Código Css De Cada Modulo Index.Php.....43
Ilustración 14. Extracción De Código Css De Cada Modulo Index.Php.....44
Ilustración 15. Llamado *Ajax* 1 Que Se Utilizaban. 45
Ilustración 16. Llamado *Ajax* 2 Que Se Utilizaban. 45
Ilustración 17. Llamados Ajax Unificado. 46
Ilustración 18. Archivo Javascript Antiguo. 47
Ilustración 19. Archivo *Javascript* Modificado. 48
Ilustración 20. Ingreso O Login Antiguo De *Sost*. 50
Ilustración 21. Ingreso O Login Modificado De Sost. 50
Ilustración 22. Recuperación De Contraseña. 51
Ilustración 23. Invalidar Sesiones Activas. 52
Ilustración 24. Menú Antes De La Actualización. 53
Ilustración 25. Menú Después De La Actualización. 53
Ilustración 26. Comparación De Interfaz En Módulos De Sacar Reportes 54
Ilustración 27. Comparación De Interfaz En Módulos De Anular Recibos. 55
Ilustración 28. Comparación De Interfaz En Módulos De Tramites 55
Ilustración 29. Comparación De Interfaz En Módulos De Crud. 56
Ilustración 30. Comparación De Interfaz En Módulo De Días Hábiles. 56
Ilustración 31. Comparación De Interfaz En Módulos De Ingresar Reportes.....57
Ilustración 32. Comparación De La Botonera En La Interfaz De Todos Los Modulos. 58
Ilustración 33. Chat En La Interfaz Antigua. 58
Ilustración 34. Chat En La Interfaz Nueva. 59
Ilustración 35. Implementación Del Método Alert(). 60

Ilustración 36. Método Original Alert() 61
Ilustración 37. Implementación Del Método Confirm (). 62
Ilustración 38. Método Original Confirm() 62
Ilustración 39. Implementación Del Metodo Prompt. 63
Ilustración 40. Método Original Prompt() 63
Ilustración 41. Codificación De *Alertify* De Negación. 65
Ilustración 42. *Alertify* De Negación Implementado. 65
Ilustración 43. Codificación De Alertify De Afirmación. 66
Ilustración 44. Alertify De Afirmación Implementado 66
Ilustración 45. Codificación De *Alertify* De Confirmación. 67
Ilustración 46. Alertify De Confirmación Implementado. 67
Ilustración 47. Codificación De Alertify De Sugerencia. 68
Ilustración 48. Alertify De *Prompt()* Implementado. 68
Ilustración 49. Loader Antiguo. 69
Ilustración 50. Loader Nuevo. 70
Ilustración 51. *Alertify* De Confirmación Implementado. 72
Ilustración 52. Archivo De Clases Sql.Php. 72
Ilustración 53. Archivo Index_Sql.Php Sin Clases. 73
Ilustración 54. Funciones A Sustituir. 74
Ilustración 55. Archivo Index_Sql.Php Con Clases. 75
Ilustración 56. Errores Comunes. 77
Ilustración 57. Capacitacion Sost 2018. 79
Ilustración 58. Cambio De Nombre De Funciones. 80
Ilustración 59. Nuevo Archivo De Clases. 81
Ilustración 60. Implementacion 82
Ilustración 61. Error Por Variables No Definidas. 83
Ilustración 61. Prueba #1 Sost Original. 86
Ilustración 62. Prueba #1 Sost Actualizada. 86
Ilustración 63. Resultados #1 Sost Original. 87
Ilustración 64. Falencias Obtenidas *Sost* Original Analisis #1. 88
Ilustración 65. Resultados #1 Sost Actualizado. 89
Ilustración 66. Analisis #2 Sost Original. 91
Ilustración 67. Línea De Tiempo Prueba #2 *Sost* Original 92
Ilustración 68. Analisis #2 Sost Actualizado. 92
Ilustración 69. Línea De Tiempo Prueba #2 Sost Actualizado. 93

LISTA DE ANEXOS

ANEXO A. Documentación de pruebas realizadas a cada módulo. 98
ANEXO B. Cronograma implementado en el proyecto. 102

RESUMEN

TITULÓ: PRACTICA EMPRESARIAL: REFACTORIZACIÓN DEL SISTEMA DE INFORMACIÓN *SOST* PARA SOPORTAR UNA ACTUALIZACIÓN TECNOLÓGICA TANTO GRÁFICA COMO FUNCIONAL, BASADO EN UN PATRÓN MVC. *

AUTOR: LUIS CARLOS ALVAREZ ALVAREZ**

PALABRAS CLAVES: MOVILIDAD Y SERVICIOS S.A.S, TRANSTIO DE GIRON, *SOST*, REFACTORIZACION.

DESCRIPCIÓN:

La empresa Movilidad y Servicios Girón s.a.s., tiene como función la administración, el recaudo de los trámites y los servicios ofrecidos por la Secretaría De Tránsito Municipal De Girón. Es una empresa a nivel nacional, con una cantidad de usuarios y flujo de trabajo constante; donde manejan un sistema de información SOST (Software Online Secretaria de Transito) para captar la información de sus solicitantes.

El área de sistemas de Movilidad y Servicios Girón s.a.s. es la encargada de proveer los servicios de desarrollo, capacitación y soporte de todos los procesos sistematizados de la empresa. El software PARE utilizado antes por la empresa no daba un buen rendimiento en los procesos. Por tal motivo la empresa tuvo la necesidad de implementar el desarrollo de un nuevo software que fuera más seguro, más formidable, y que contenga nuevos módulos y procesos que ayuden con el desarrollo de las actividades. En el 2013 fue creado el software SOST la herramienta más importante del tránsito de Girón, donde implementa un diseño modular.

Desde su lanzamiento el software SOST maneja una estructura con lenguajes de programación que en su época eran las versiones actuales. Hoy en día surgieron versiones nuevas donde Se Encuentran clases y funciones obsoletas y desactualizadas. A su vez la organización de la codificación de los lenguajes de programación está estructurada de tal manera que no manejan un patrón Modelo-Vista-Controlador.

El presente proyecto consiste en actualizar la interfaz gráfica, reestructurando las funciones y clases obsoletas, para soportar una actualización tecnológica a nivel de los lenguajes de programación. Durante el proceso de desarrollo del proyecto se analizará el código fuente, se manipulará su estructura, se buscarán las actuales versiones de lenguajes de programación permitidas para el funcionamiento del software. Con un conjunto de pruebas se verifica el correcto funcionamiento de la nueva estructura.

---

* Trabajo de Grado

** Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Fernando Ruiz Diaz. Ingeniería de Sistemas.

ABSTRACT

TITLE: EMPRESARIAL PRACTICE: REFACTORIZATION OF THE *SOST* INFORMATION SYSTEM TO SUPPORT A TECHNOLOGICAL UPGRADING, BOTH GRAPHIC AND FUNCTIONAL, BASED ON A MVC PATTERN. *

AUTHOR: LUIS CARLOS ALVAREZ ALVAREZ **

KEYWORDS: MOBILITY AND SERVICES S.A.S, TRANSTIO DE GIRON, *SOST*, REFACTORING

DESCRIPTION:

The company Movilidad y Servicios Girón has the function of administration, the collection of procedures and the services offered by the Municipal Traffic Secretariat of Girón. It is a nationwide company, with a constant number of users and workflow; where they manage a SOST information system (Software Online Transit Secretary) to capture the information of their applicants.

The area of systems of Mobility and Services Girón s.a.s. is the burden of providing development, training and support services for all the systematized processes of the company. The software PARE used before by the company did not give a good performance in the processes. For what reason the company had the need to implement the development of new software that was safer, more formidable, and that contains new modules and processes that help with the development of activities. In 2013, the SOST software was created as the most important transit tool in Girón, where it implements a modular design.

Since its launch, the SOST software manages a structure with programming languages that in their time are the current versions. Today new versions emerged where obsolete and outdated classes and functions are found. In turn, the organization of coding of programming languages is structured in such a way that a Model-View-Controller model can not be controlled.

The present project consists of updating the graphical interface, restructuring obsolete functions and classes, to support a technological update at the level of the programming languages. During the development process of the project, the source code is analyzed, its structure is manipulated, the current versions of programming languages allowed for the operation of the software are searched. With a set of tests the correct functioning of the new structure is verified.

_____

* Degree work

** Faculty of Physical-Mechanical Engineering. School of Systems and Information Engineering. Director: Fernando Ruiz Diaz. Systems engineer.

INTRODUCCIÓN

Actualmente se está en una época donde lo sistematizado es tan fundamental para una entidad o empresa donde depende el desarrollo, el avance y la integridad que la empresa tiene con la información administrada, la información se ha situado en uno de los principales recursos que poseen las empresas actualmente. Los entes que se encargan de las tomas de decisiones han comenzado a comprender que la información no es sólo un subproducto de la conducción empresarial, sino que a la vez alimenta a los negocios y puede ser uno de los tantos factores críticos para la determinación del éxito o fracaso. Por ello se opta por utilizar los sistemas de información (SI), que permiten:

- Lograr Automatizar los procesos operativos,
- Proporcionar información de apoyo a la toma de decisiones,
- Lograr ventajas competitivas a través de su implantación y uso.

Por ello Movilidad y Servicios Girón S.A.S empresa donde se está desarrolla el proyecto de grado, maneja un sistema de información denominado *SOST* (Software Online Secretaria de Transito) el cual es parte fundamental de sus procesos, donde se abarcan 173 módulos destinados a facilitar el desarrollo de las actividades empresariales. Por tal motivo el tema de este proyecto se centrará en el software *SOST* donde se realizará una refactorización a la interfaz gráfica, código, clases y funciones obsoletas que impiden el avance tecnológico y actualizable de las versiones (*PHP, APACHE, MYSQL*) a utilizar.

El desarrollo de este proyecto pretende soportar una actualización tecnológica que se ha quedado atrasado a nivel de versiones de lenguajes de programación actualmente utilizados en el software (*PHP, MYSQL, APACHE*), con mayor rapidez, seguridad y escalabilidad en el software.

Este documento se generalizarán los ejemplos, escogiendo los módulos que se destacan para esclarecer los cambios y pruebas realizadas, también presenta al

lector de manera organizada por capítulos, donde encontramos las generalidades, en la cual se especifican el planteamiento y justificación del problema, los objetivos alcanzados, así como el marco teórico y metodológico, los resultados y conclusiones.

ILUSTRACIÓN 1. INSTALACIONES MOVILIDAD Y SERVICIOS GIRON S.A.S

1. MOVILIDAD Y SERVICIOS GIRÓN S.A.S.

Es una empresa de economía mixta conformada por el municipio de San Juan Girón y la Empresa Growing Network S.A, la cual tiene como función la administración y recaudo de los trámites y servicios ofrecidos por la Secretaria De Tránsito Municipal De Girón y la administración en el recaudo de los impuestos correspondientes a la Secretaria De Hacienda Municipal De Girón. Creada el 14 de diciembre del año 2011, con el objetivo de gestionar la administración y recaudo de los trámites y servicios ofrecidos por la Secretaria De Tránsito Municipal De Girón

Donde se fortalece la plataforma de software, desarrollando aplicaciones a la medida y mejoramiento de la calidad, productividad y eficacia en los procesos organizacionales. Se perfila como un tránsito líder en la región, resultado del mejoramiento continuo y del afán por ofrecer un buen servicio.

Consta de nuevas instalaciones, más cómodas, seguras y amplias, ofreciendo una plataforma tecnológica de último avance, optimizando los recursos tanto operativos como tácticos.

1.1 MISIÓN:

Movilidad y Servicios Girón S.A.S tiene como misión la prestación de servicios de soporte operativo y tecnológico, especializado en el diseño e implementación de soluciones administrativas, a través del uso de tecnología de punta y personal altamente calificado con sentido de responsabilidad, eficiencia y confiabilidad, satisfaciendo las necesidades de nuestros clientes y usuarios.[1]

1.2 VISIÓN:

En el año 2022, Movilidad y Servicios Girón S.A.S será una entidad líder en el desarrollo de soluciones tecnológicas y operativas, posicionando los servicios ofertados por la entidad, como los mejores del país, basados en una cultura de calidad y mejoramiento continuo.

[1] MOVILIDAD Y SERVICIOS GIRON S.A.S. (2012). Tránsito de girón: nosotros [en línea] Disponible en: <http://www.transitodegiron.com.co/> [citado 14 de diciembre de 2011]

2. OBJETIVOS

2.1 OBJETIVO GENERAL:

Diseñar e implementar una nueva estructura de los artefactos funcionales y gráficos del sistema *SOST* para soportar una actualización tecnológica.

2.2 OBJETIVOS ESPECÍFICOS:

- Diagnosticar la interfaz gráfica actualmente utilizada restructurando las funciones y clases obsoletas para la actualización.
- Unificar las clases, funciones y procesos para consolidar el uso del patrón Modelo-vista-controlador (MVC) de la aplicación existente.
- Implementar la funcionalidad para el soporte de la estructura definida derivada del diagnóstico inicial.
- Validar a través de un conjunto de pruebas que garanticen el correcto funcionamiento del aplicativo con la nueva estructura.
- Documentar las modificaciones y las pruebas realizadas en el aplicativo.

3. MARCO TEÓRICO

3.1 CONTEXTUALIZACIÓN DEL PROYECTO

Se presentarán las bases teóricas en las que se fundamenta cada una de las fases del proyecto. Partiendo de esta información se desarrolló e implementó el desarrollo del proyecto.

3.1.1 Análisis. Como primera instancia se realizó un estudio del software *SOST* donde se desarrolla el proyecto, este estudio empezó con los antecedentes que ha tenido este software al día de hoy, se encontraron con lenguajes de programación y servidores desactualizados, cuando fue creado en su época se implementaron las últimas versiones. Hoy en día ya se han desarrollado nuevas versiones de lenguajes de programación y servidores los cuales han adaptado más funciones y nos aseguran la integridad de los datos.
Se realiza un análisis del código fuente del software *SOST* donde se evidencia que fragmentos de código pueden ser mejorados para bien, eliminando código y reemplazándolo por inclusión de rutas; necesarias para que inicialicen correctamente los módulos, también descartando funciones y clases obsoletas que puedan interrumpir la actualización de las nuevas versiones de los lenguajes de programación.

SOST maneja varios tipos de lenguajes de programación los cuales se encuentran homogéneamente implementados, Un ejemplo son las funciones *JavaScript* que están dentro de archivos que contenían la estructura HTML del respectivo modulo, así como las funciones de base de datos que realizan llamados de forma directa y no están predefinidas por clases en su totalidad y minimizando la posibilidad de cambiar de conexión o versión con una mayor facilidad.

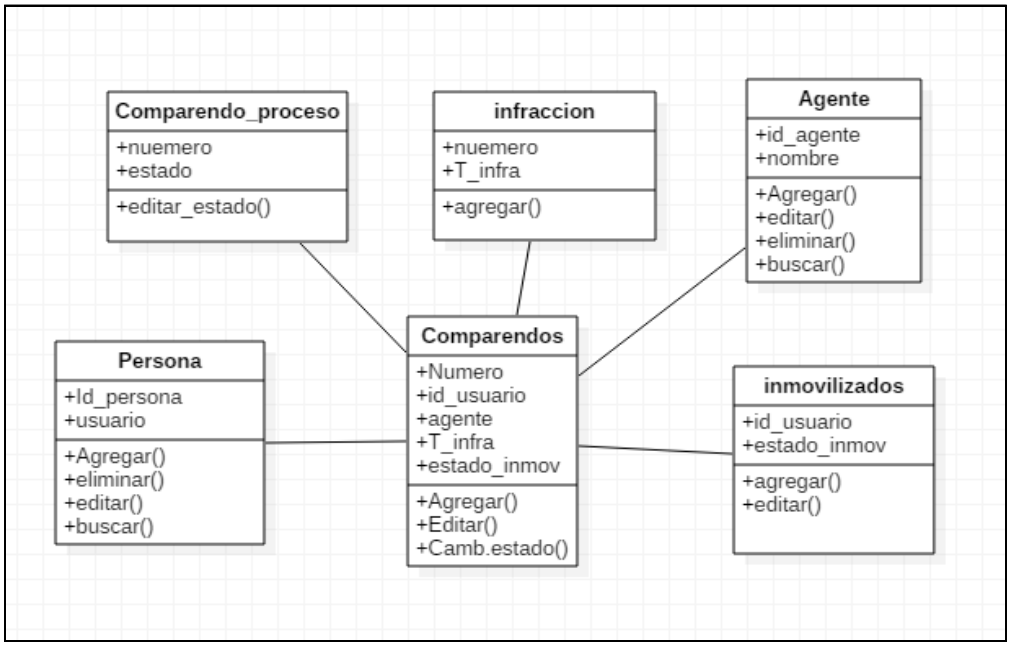
En el tema visual tiene aspectos antiguos o básicos donde casi no se implementan estilo con css, utilizan *Alert*as de falta de información o de condiciones que utilizan algunos módulos los cuales son nativos del lenguaje de programación.
Con esta información recolectada se procede a realizar los respectivos cambios tanto en la estructura funcional como gráfica teniendo en cuenta que el resultado del proceso no se va alterar.

3.1.1.1 Análisis Estructural. Se estudió como estaba compuesto el software, como era su organización y como era su funcionalidad, el *SOST* es un software muy amplio por su cantidad de módulos y funciones. Se van a realizar los diagramas de clases (UML) resaltando los procesos más importantes que la empresa ofrece.

- **Diagrama de clases (UML):**

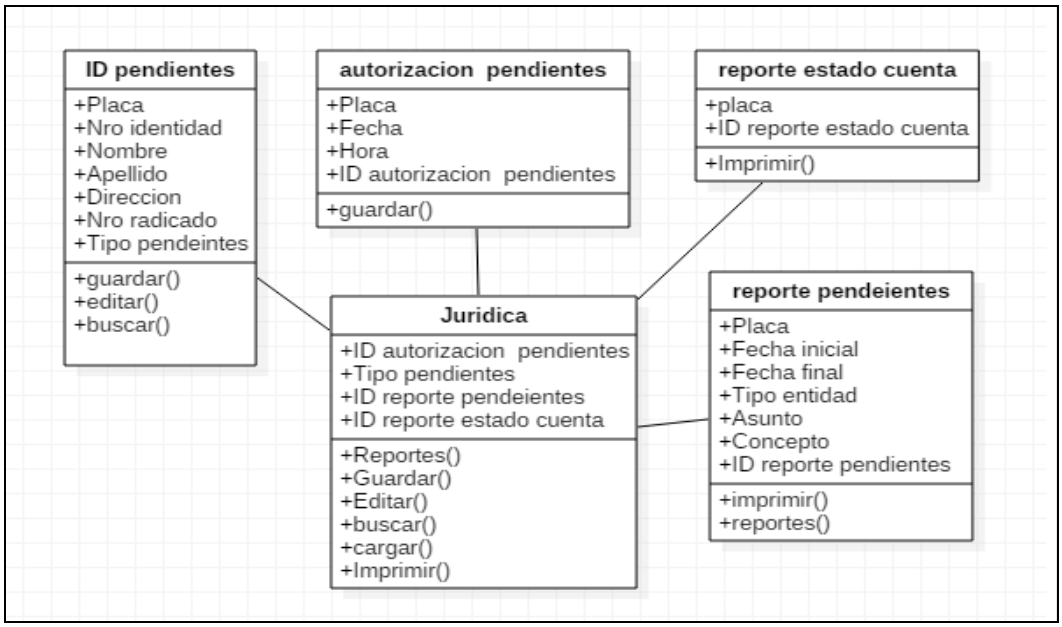
El diagrama UML más comúnmente usado, y la base principal de toda solución orientada a objetos. Los diagramas de clases describen la estructura estática de un sistema. Las cosas que existen y que nos rodean se agrupan naturalmente en categorías. Una clase es una categoría o grupo de cosas que tienen atributos (propiedades) y acciones similares.

FIGURA 1. DIAGRAMA CLASES (UML) - PROCESO COMPARENDOS.



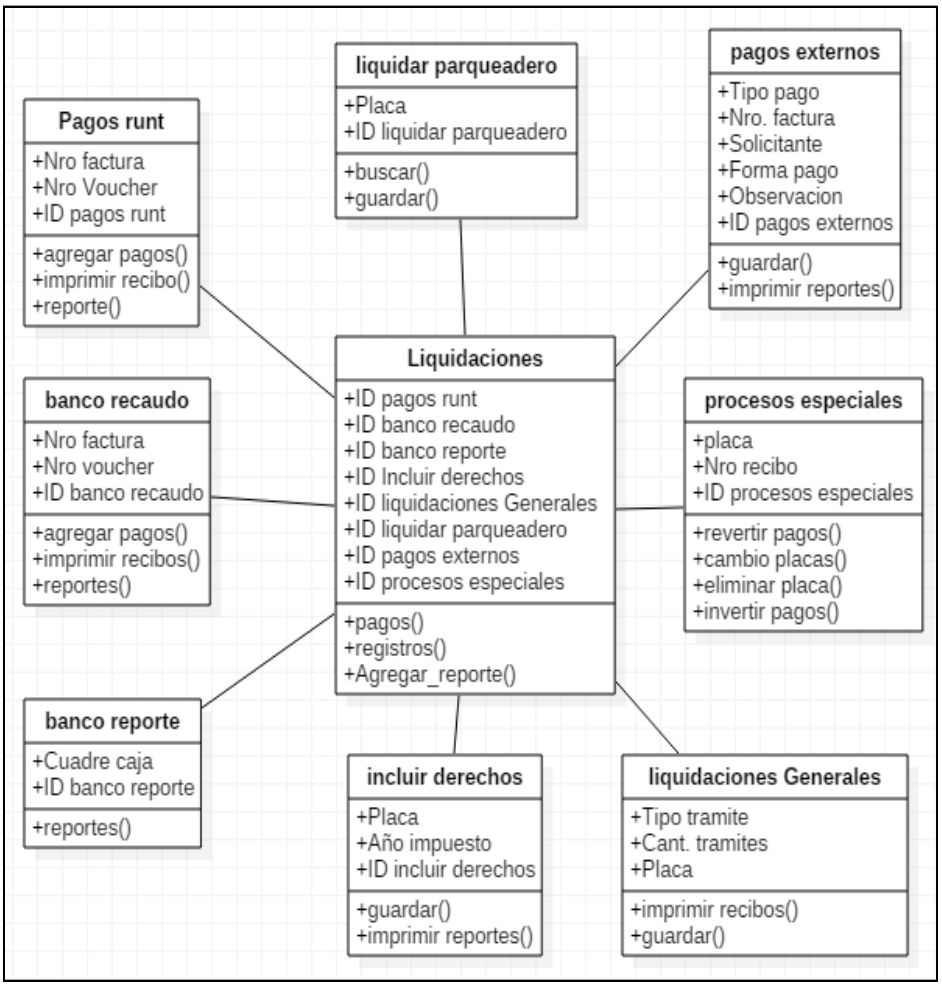
El proceso de comparendos tiene funciones de agregar, eliminar, editar buscar y cambiar de estado, este proceso está compuesto por cinco módulos los cuales ayudan a complementar el proceso.

FIGURA 2. DIAGRAMA CLASES (UML) - PROCESO JURIDICA.



El proceso de jurídica tiene las funciones de generar reportes, imprimir reportes, editar, buscar, así como el control de inscripción y levante de medidas cautelares (embargos, des-embargos, etc.), ya que es un proceso delicado, solo lo manipulan abogados y personas con el nivel profesional de llevar a cabo el proceso, está compuesto por cuatro módulos las cuales complementan el proceso.

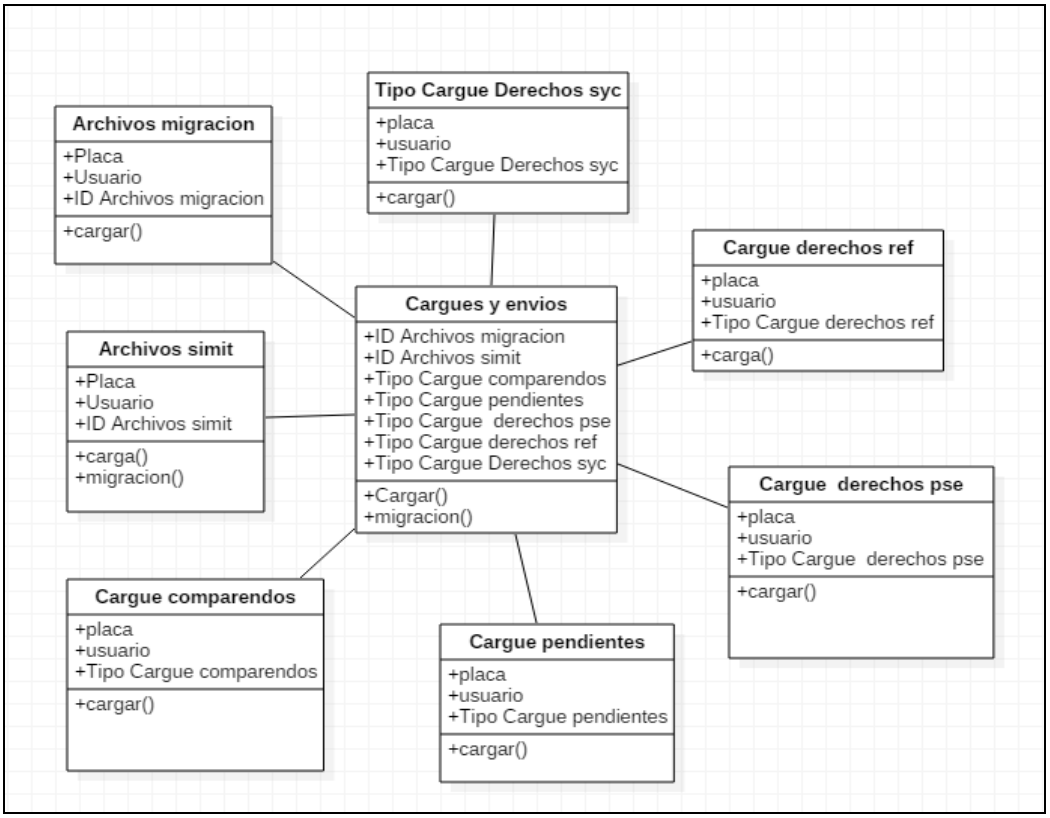
FIGURA 3. DIAGRAMA CLASES (UML) - PROCESO LIQUIDACIONES.



El proceso de liquidaciones tiene las funciones de pagos, registrar facturas, imprimir recibos, cargar pagos, guardar e invertir pagos al igual que lo procesos anteriores al ser uno de los procesos más importantes, las personas que manejan

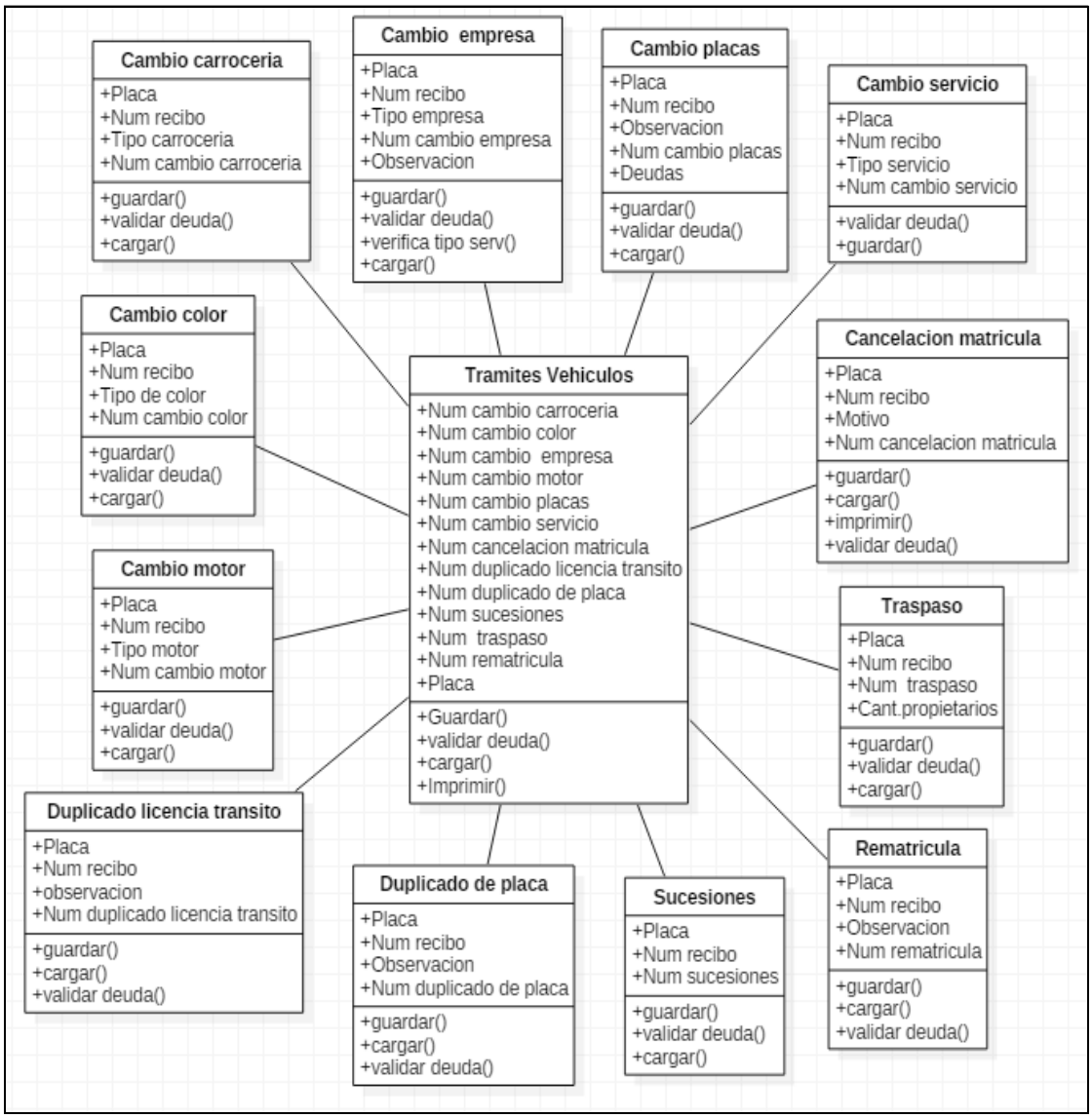
los módulos para culminar el proceso tienen una gran responsabilidad, ya que son los ingresos que obtiene la empresa, donde estos módulos lo manejan el banco responsable a recibir el pago de los recibos.

FIGURA 4. DIAGRAMA CLASES (UML) - PROCESO CARGUES Y ENVIOS.



El proceso de cargues y envíos contiene funciones de cargar y de migración donde este proceso hace que la información sea enviada a los módulos indicados para su buena implementación, también donde exportan la información requerida para mostrar reportes, informes y documentos que soporten los tramites y procesos llevados a cabo en la empresa por medio del software *SOST*, estos módulos son manipulados por funcionarios del tránsito quienes son responsables de la integridad de la información requerida.

FIGURA 5. DIAGRAMA CLASES (UML) - PROCESO TRAMITES.



El proceso de tramites de vehículos contiene funciones como: guardar, validar_deuda, cargar e imprimir; se puede decir que este proceso es el que más se ejecuta en la empresa pues las necesidades de los usuarios lo demuestran, este proceso es el más grande y uno de los más importantes pues como su nombre lo indica es el que lleva a cabo los tramites vehiculares, este proceso está conformado por una gran cantidad de módulos que lo complementan el proceso de los tramites y es manipulado por los funcionarios de la empresa.

3.1.2 Refactorización.

3.1.2.1 Modelo vista controlador (m.v.c). Modelo Vista Controlador (M.V.C) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo[2].

- El Modelo que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.
- La Vista, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.
- El Controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

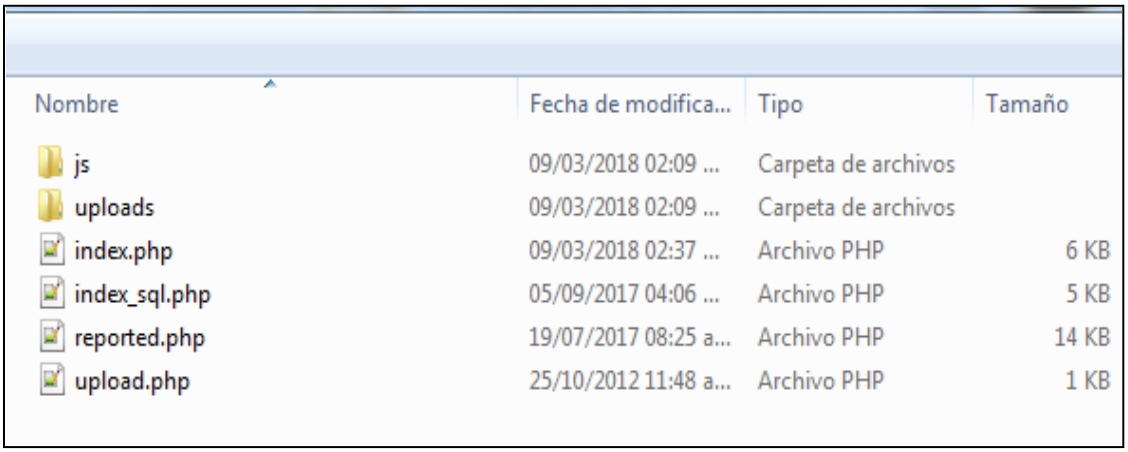
Surge de la necesidad de reestructurar el software *SOST*, ofreciendo una solución más robusta, con un ciclo de vida más adecuado y una escalabilidad constante, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos. Se implementó el modelo-vista-controlador ya que algunos de los módulos como se mencionó anterior mente tienen homogeneidad en sus archivos, como hay módulos que al tener el (M.V.C) tienen más archivos ya que generan reportes y cartas que no se pueden excluir un ejemplo visual de cómo se encontró la organización de los respectivos archivos.

La mayoría de módulos se encontraron con dos archivos frecuentes, los cuales son index.Php este contenía la estructura HTML, estilos con código css y

[2] UNIVERSIDAD DE ALICANTE. Servicio de Informática ASP.NET MVC 3 Framework: Modelo vista controlador (MVC) [en línea] Disponible en: < https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html >.

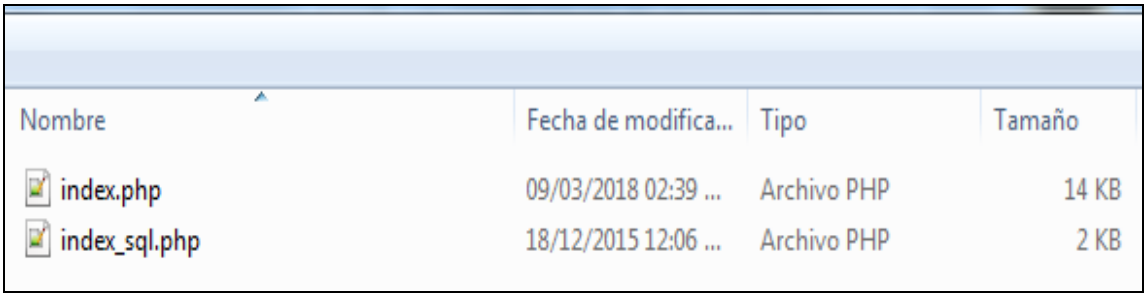
funciones *JavaScript* y el index_sql.Php que contiene las consultas *MySQL* por medio de código Php.

ILUSTRACIÓN 2. ARCHIVOS (Modulo reporte data crédito).



se puede observar que este módulo está compuesto por 2 carpetas y 4 archivos donde no se está manejando el (M.V.C) además hay archivos que no son funcionales.

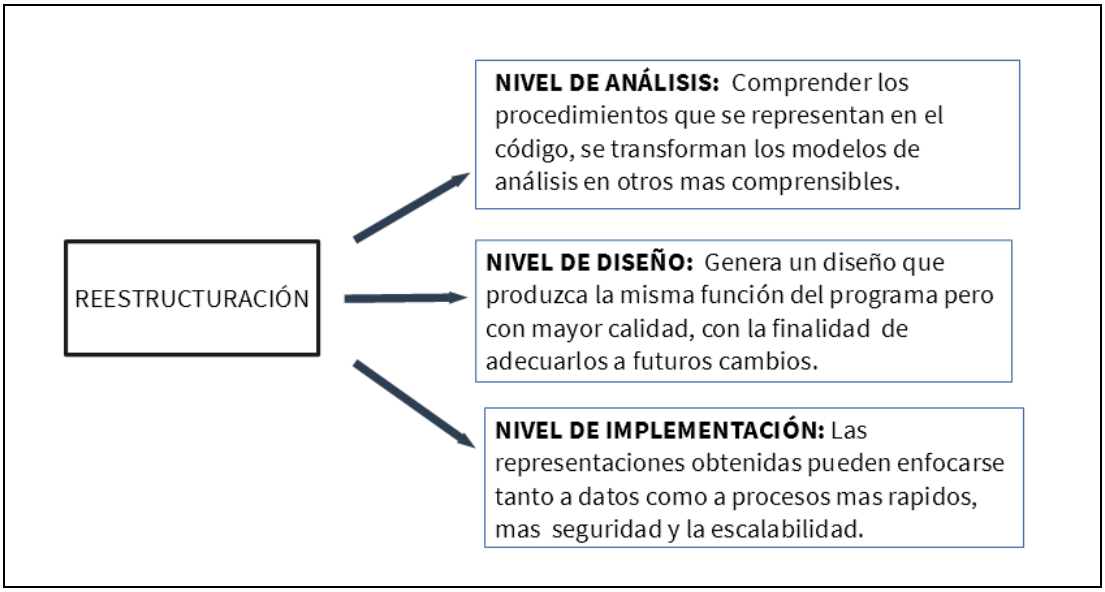
ILUSTRACIÓN 3. ARCHIVOS (Modulo archivos para archivar).



se puede observar que solo se encuentran 2 archivos y no se utiliza el (M.V.C) una gran mayoría de archivos están compuestos de esta manera.

3.1.2.2 Reestructuración. Es el reordenamiento, la reorganización o modificación de determinado tipo de estructuras en ámbitos y espacios específicos se aplicó este término al software ya que está compuesto de estructuras agrupadas en un mismo archivo.

FIGURA 6. IMPLEMENTACIÓN DE LA REESTRUCTURACIÓN.

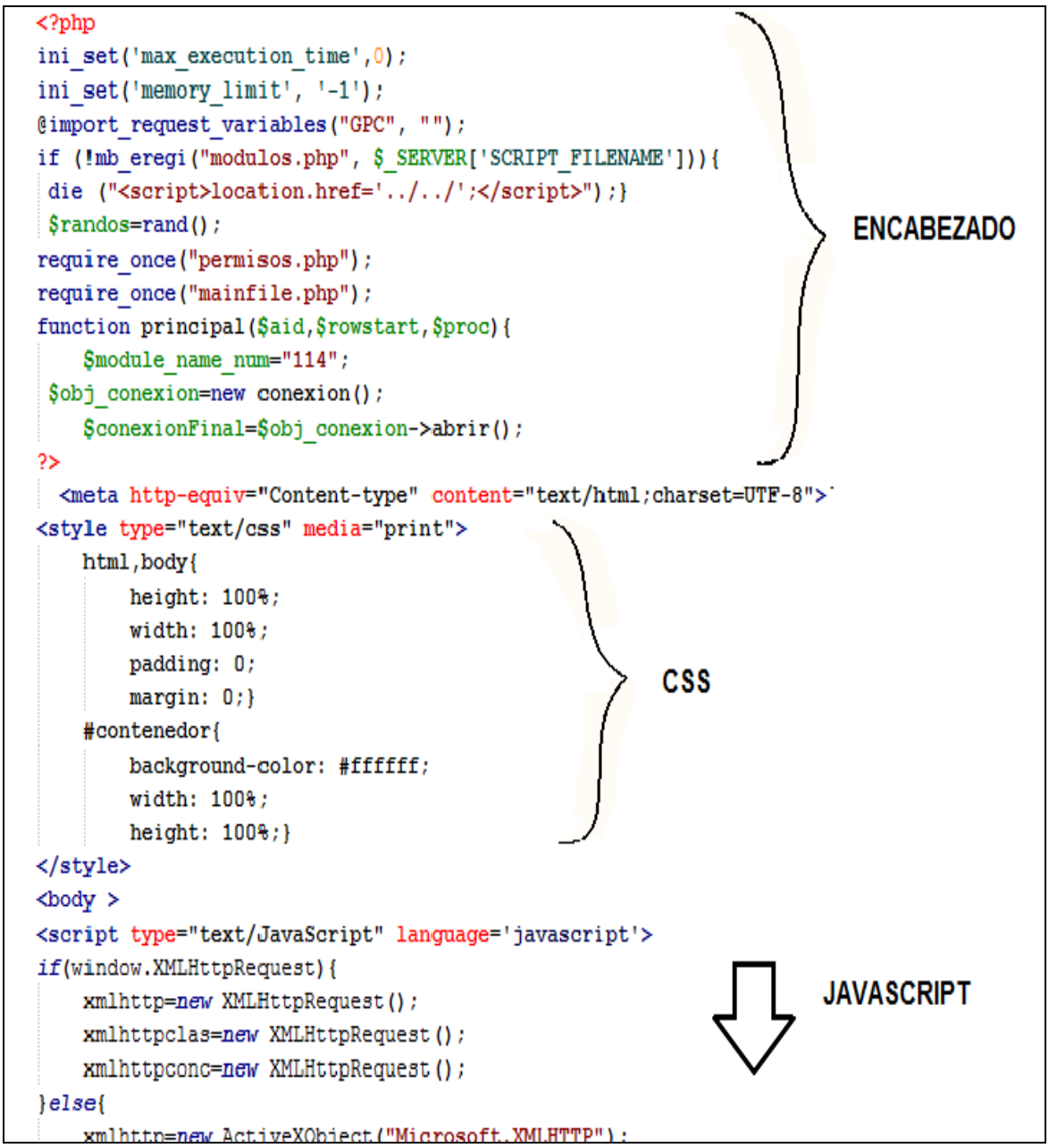


Se realizó el respectivo estudio de bloques de cogido que se podían simplificar por medio de inclusiones de rutas, todos los encabezados de cada módulo se reestructuraron, también se hizo la exclusión de funciones *JavaScript* contenidas en archivos HTML llamados index.Php el cual se destinó como la vista del (M.V.C) y donde se crea un nuevo archivo llamado funsiones.js donde va ser el controlador del (M.V.C) el cual además de manejar funciones maneja también el flujo de datos y validaciones por medio de *Ajax*.

El archivo index_sql.Php maneja las consultas *MySQL* por medio de lenguaje de programación Php las cuales tienen rutas muy extensas donde involucra 2 o 3

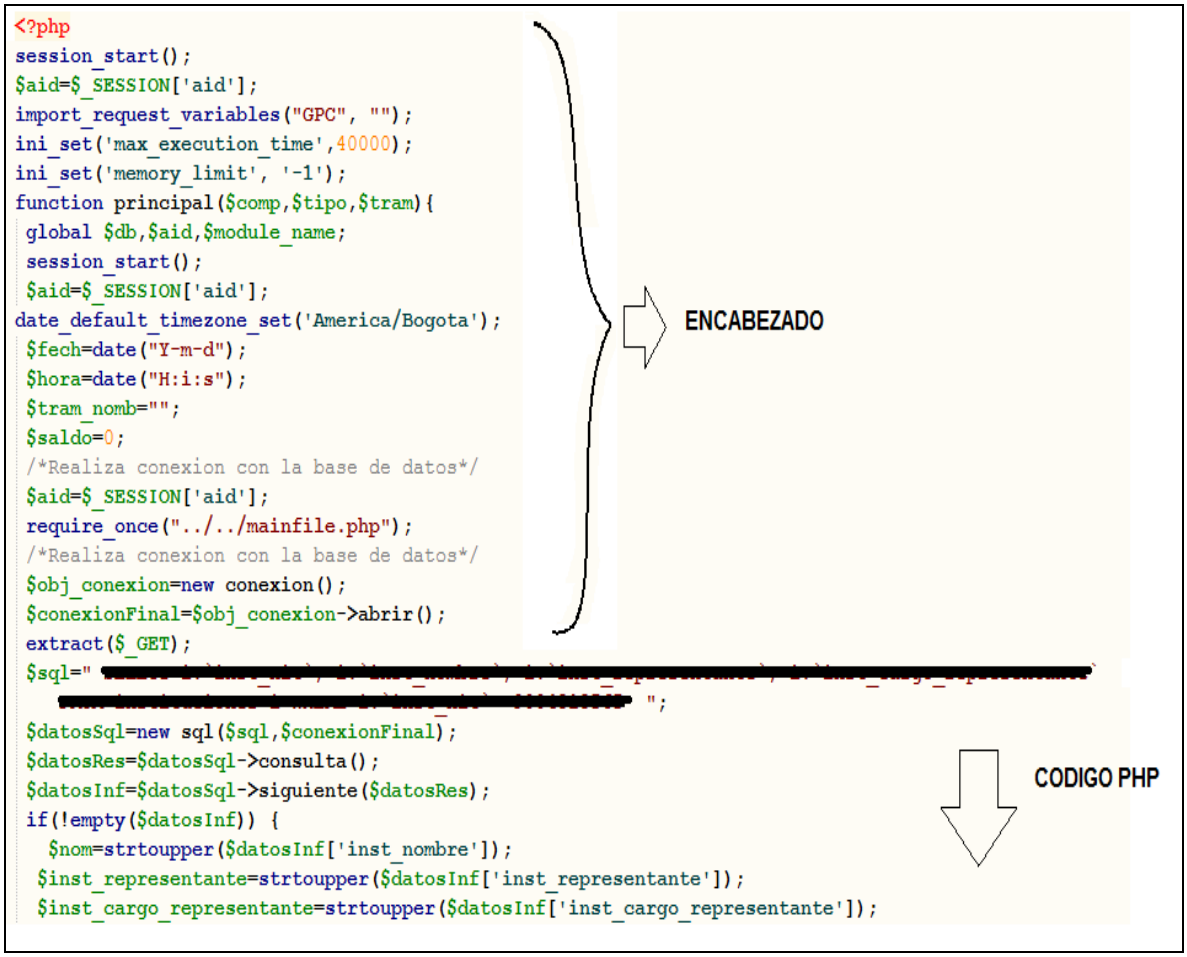
niveles de carpetas para llegar al archivo ideal se simplifica esta ruta de manera que no afecte el funcionamiento, este archivo sería el modelo del (M.V.C).

ILUSTRACIÓN 4. CÓDIGO FUENTE INDEX.PHP




```
<?php
ini_set('max_execution_time',0);
ini_set('memory_limit', '-1');
@import_request_variables("GPC", "");
if (!mb_eregi("modulos.php", $_SERVER['SCRIPT_FILENAME'])){
 die ("<script>location.href='../../';</script>");}
 $randos=rand();
require_once("permisos.php");
require_once("mainfile.php");
function principal($aid,$rowstart,$proc){
    $module_name_num="114";
 $obj_conexion=new conexion();
    $conexionFinal=$obj_conexion->abrir();
?>
  <meta http-equiv="Content-type" content="text/html;charset=UTF-8">
<style type="text/css" media="print">
    html,body{
        height: 100%;
        width: 100%;
        padding: 0;
        margin: 0;}
    #contenedor{
        background-color: #ffffff;
        width: 100%;
        height: 100%;}
</style>
<body >
<script type="text/JavaScript" language='javascript'>
if(window.XMLHttpRequest){
    xmlhttp=new XMLHttpRequest();
    xmlhttpclas=new XMLHttpRequest();
    xmlhttpconc=new XMLHttpRequest();
}else{
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
```

ILUSTRACIÓN 5. CÓDIGO FUENTE INDEX_SQL.PHP




```php
<?php
session_start();
$aid=$_SESSION['aid'];
import_request_variables("GPC", "");
ini_set('max_execution_time',40000);
ini_set('memory_limit', '-1');
function principal($comp,$tipo,$tram){
 global $db,$aid,$module_name;
 session_start();
 $aid=$_SESSION['aid'];
date_default_timezone_set('America/Bogota');
 $fech=date("Y-m-d");
 $hora=date("H:i:s");
 $tram_nomb="";
 $saldo=0;
 /*Realiza conexion con la base de datos*/
 $aid=$_SESSION['aid'];
 require_once("../../mainfile.php");
 /*Realiza conexion con la base de datos*/
 $obj_conexion=new conexion();
 $conexionFinal=$obj_conexion->abrir();
 extract($_GET);
 $sql="
    ";
 $datosSql=new sql($sql,$conexionFinal);
 $datosRes=$datosSql->consulta();
 $datosInf=$datosSql->siguiente($datosRes);
 if(!empty($datosInf)) {
   $nom=strtoupper($datosInf['inst_nombre']);
  $inst_representante=strtoupper($datosInf['inst_representante']);
  $inst_cargo_representante=strtoupper($datosInf['inst_cargo_representante']);
```

Se debe aclarar que la información oculta en esta imagen esta tachada debido a políticas de seguridad de la empresa e integridad de la información.

Las dos ilustraciones, la ilustración 4 e ilustración 5 es la estructura principal que se encontró al hacer el análisis ya que, cada módulo contiene su respectiva carpeta y por dentro de cada carpeta están estos dos principales archivos como los observamos en la ilustración 2 y la ilustración 3.

3.2 HERRAMIENTAS UTILIZADAS EN EL DESARROLLO DEL PROYECTO

3.2.1 Sost. (Software Online Secretaria de Transito) es el software más importante que se maneja a nivel de la empresa ya que consta de 173 módulos los cuales realizan diferentes tareas o funciones para el desarrollo del día a día de la empresa debido a la demanda de procesos que ejerce la empresa. Maneja las liquidaciones, tramites de los vehículos (licencias, tramites y matriculas) también generan reportes diarios y mensuales de ingresos económicos. En el tema gráfico hay muchos módulos que su interfaz es semejante a otras, pero su funcionalidad es completamente distinta entre todos los módulos.

El software *SOST* se basa en los lenguajes: *PHP, JAVASCRIPT, HTML, CSS, JQUERY y AJAX* y como motor de base de datos *MYSQL*; Las cuales son herramientas de código abierto que ayudan a cumplir con el objetivo que es gestionar las liquidaciones de Derechos Municipales para el municipio de Girón y tramites de los vehículos (licencias, tramites y matriculas) también generan reportes diarios y mensuales de ingresos.

3.2.1.1 Apache. Es un servidor web de código abierto que implementa el protocolo HTTP/1.1, funciona en Windows 2000, NetWare 5.x y versiones posteriores, OS/2 y la mayoría de versiones de Unix.[3]
Establece funciones solicitadas con frecuencia que incluyen: Las bases de datos DB también, como bases de datos relacionales y LDAP (Protocolo Ligero de Acceso a Directorios) para la autenticación, respuesta personalizadas a errores y problemas, Registros configurados y confiables: puede configurar Apache para

[3]APACHE HTTP SERVER. (1997) ¿Qué es apache? [en línea] Disponible en: <https://wiki.apache.org/httpd/FAQ#What_is_Apache.3F >.

generar registros en el formato que desee. Además, en la mayoría de las arquitecturas de Unix.

3.2.1.2 Php. (Procesador de hipertexto) Es un lenguaje de código abierto de propósito general de código del lado del servidor muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML[4].

3.2.1.3 JAVASCRIPT. Es un lenguaje de programación interpretado, se define como orientado a objetos basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas todos los navegadores modernos[5].

3.2.1.4 Html. (Lenguaje de marcas de hipertexto) Es el elemento de construcción más básico de una página web y se usa para crear y representar visualmente una página web. Determina el contenido de la página web, pero no su funcionalidad.

3.2.1.5 Css. (Hojas de estilo en cascada) Es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje en XML (Lenguaje de Marcado Extensible) como son HTML. Es muy usado para establecer el diseño visual de los documentos web.

3.2.1.6 Jquery. Es una biblioteca multiplataforma de *JavaScript* rápida, pequeña y rica en funciones. Hace cosas como el recorrido y manipulación de documentos HTML, manejo de eventos, animación con la técnica *AJAX* a páginas web, jQuery es la biblioteca de *JavaScript* más utilizada.

[4] PHP (2001). Manual Php: Conceptos básicos. [en línea] Disponible en: <http://php.net/manual/es/intro-whatis.php >.
[5] EORTS (2017). Programación: JavaScript. [en línea] Disponible en: < http://eorts.com/javascript.html>.

3.2.1.7 Ajax. Es el acrónimo de (Asíncronos *JavaScript* and XML) es decir (*JavaScript* y XML Asíncrono) es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor, se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página se define como una técnica para el desarrollo de páginas web que implementan aplicaciones interactivas[6].

3.2.1.8 Mysql. Es un sistema de gestión de base de datos relacional de código abierto más popular del mundo donde consta de su rendimiento comprobado, fiabilidad y facilidad de uso, desarrollado bajo licencia dual GPL (Licencia pública general) y licencia comercial[7].

3.2.2 Hardware

3.2.2.1 Hpe proliant ml110 hardware. El servidor HPE ProLiant ML110 es compacto con menos de 19 pulgadas de profundidad, silencioso, accesible y ofrece el rendimiento necesario para las exigencias de computación de MOVILIDAD Y SERVICIOS GIRON S.A.S. Contiene:

- procesadores Intel® Xeon® diseñados específicamente para servidores con velocidades de hasta 3,5 GHz y hasta 14 núcleos.
- Incluye cinco ranuras PCIe, ocho unidades de disco de factor de forma grande (LFF) o dieciséis de factor de forma reducido (SFF) y ocho DIMM DDR4 para las necesidades de las empresas en crecimiento.
- Unidad de disco duro SAS/SATA de 10 TB[8].

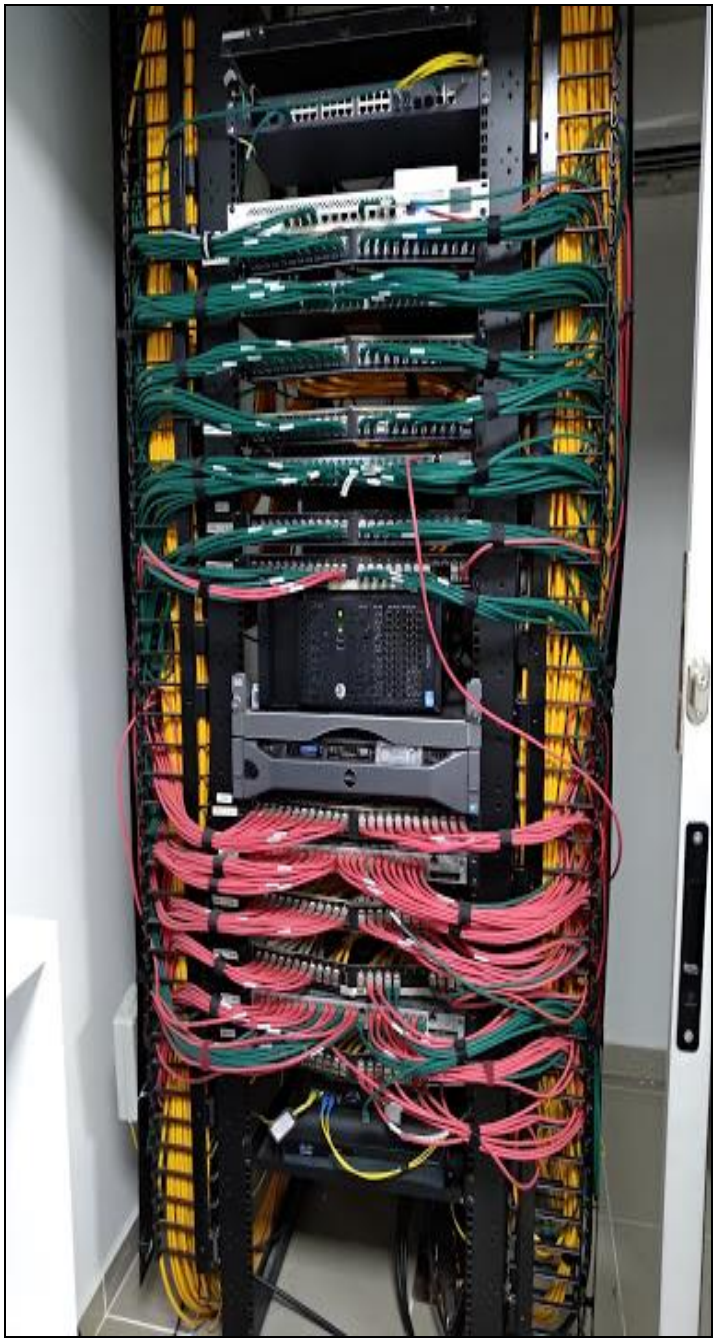
---

[6] LIBROSWEB (2006). Introducción a AJAX: Capítulo 1. Introducción a AJAX. [en línea] Disponible en: < http://librosweb.es/libro/ajax/capitulo_1.html>.

[7]WIKIPEDIA (2018). MySQL [en línea] Disponible en: <https://es.wikipedia.org/wiki/MySQL>.

[8] HPE (2018). SERVIDORES: Servidores ProLiantServidor: HPE ProLiant ML110 Gen10 [en línea] Disponible en:<https://www.hpe.com/us/en/product-catalog/servers/proliant-servers/pip.hpe-proliant-ml110-gen10-server.1010192782.html>.

ILUSTRACIÓN 6. HPE PROLIANT ML110



4. MARCO METODOLÓGICO

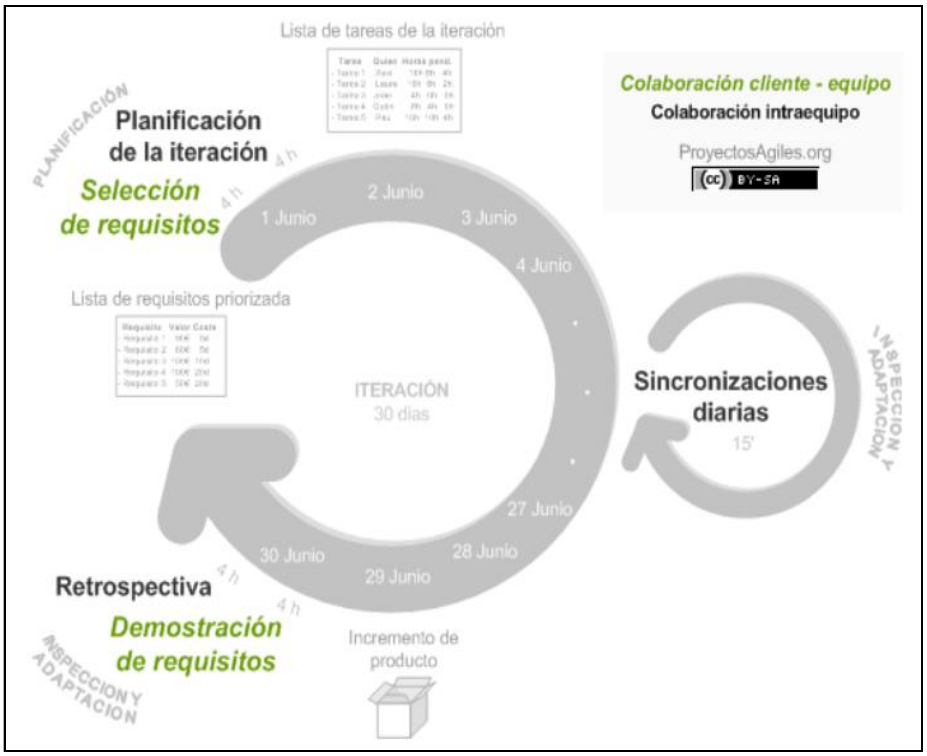
Para lograr el objetivo de caracterizar y analizar en forma integral la situación de actualizar el software *SOST* se conforma de fases, cada una de las cuales se puede dividir en bloques o tareas, las cuales guían a los desarrolladores de sistemas para que elijan las técnicas más apropiadas en cada etapa del proyecto; también facilitan la planificación, gestión, control y evaluación del mismo.

Para la implementación de actualizar la interfaz gráfica, funciones y clases de un sistema de información (*SOST*) se llevará a cabo una metodología ágil llamada "Scrum" ya que consiste en entregas parciales y regulares antes de llegar a una entrega final. Según las etapas de la metodología, el desarrollo del proyecto, el plan de trabajo a seguir utilizando es la metodología Scrum[9], la cual se ejecuta en bloques temporales cortos y fijos donde, se basa en iteraciones que normalmente son de dos semanas hasta cuatro semanas límite máximo de realimentación y reflexión, Cada bloque o iteración conlleva una tarea que hace que el proyecto se vaya desarrollando de manera eficiente y óptima. Una vez empezado y terminado el bloque de tareas se entra en una etapa Inspección y adaptación, donde son reuniones con el ingeniero a cargo del proyecto, quien determinan si las acciones realizadas están bien hechas, verificar alguna anomalía y por último aprobar el bloque de tareas.

Una vez aclarando la metodología se trabajará directamente en el servidor, con una ruta distinta a la original que utiliza *SOST* en la empresa, dicha ruta lleva a una copia de los archivos del software donde se realizarán los cambios y pruebas pertinentes, falta resaltar que se trabajara con una copia de una base de datos por la seguridad de la empresa, el cual una de las principales características de la empresa es la integridad de la información.

[9] PROYECTOS AGILES. Qué es SCRUM [en línea] Disponible en:< https://proyectosagiles.org/que-es-scrum/>.

ILUSTRACIÓN 7. METODOLOGÍA SCRUM



4.1 TAREA 1: ANÁLISIS.

4.1.1 Adaptación. Como se mencionó anteriormente se hizo un análisis extenso y complejo del software ya que es la herramienta fundamental de la empresa, no se puede tener ninguna falla o borrar alguna línea de código que ponga en duda la integridad de los datos del sistema de información.

4.1.2 Planificación de la iteración. La idea principal a desarrollar en este bloque del proyecto es establecer que cambios se deben, se pueden y cuales no se pueden realizar debido que su codificación ha pasado por varias versiones del lenguaje de programación lo que significa que hay clases y funciones obsoletas para soportar la actualización tecnológica.

4.1.3 Ejecución de la iteración. Se estudió los componentes, archivos, lenguajes de programación, estructuras donde se llegan a conclusiones válidas para lograr los objetivos del proyecto así el software podrá ser actualizado en sus versiones de lenguajes de programación logrando cambios tanto en la interfaz gráfica como la reestructuración de sus archivos utilizando el modelo-vista-controlador (M.V.C).

4.1.4 Inspección y adaptación. Los cambios a realizar se van a dividir en bloques los cuales serán:

- Organización de archivos y desagregación del cogido fuente según el (M.V.C).
- Aadaptación de interfaz general y plugin de *Alert*as (informativas)
- Aadaptación de clases en los llamados *MySQL*.
- Pruebas de funcionamiento de cada módulo sin implementar las nuevas versiones.
- Pruebas de funcionamiento y velocidad implementando las nuevas versiones.

4.2 TAREA 2: ORGANIZACIÓN DE ARCHIVOS Y COGIDO FUENTE SEGÚN EL (M.V.C).

4.2.1 Adaptación. Adaptando el Modelo-Vista-Controlador se modificarán los archivos de tal manera que toca agregar archivos para poder separar la vista de los controladores ya que se encuentras unificados en un solo archivo llamado index. También se ordenará los encabezados de cada módulo simplificando la cantidad de líneas por medio de inclusiones de rutas donde se encuentran dichos llamados ya sean estilos css, inciso de sesión, conexión a la base de datos y rutas de clases a utilizar en el módulo.

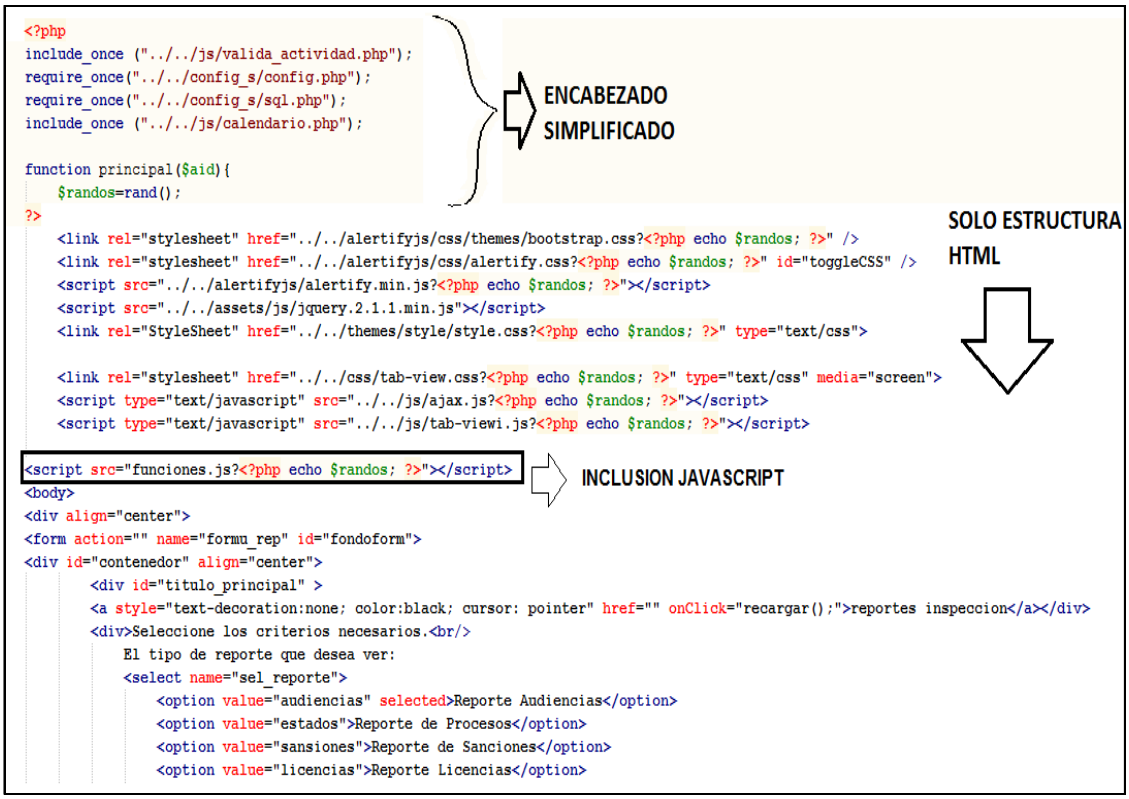
4.2.2 Planificación de la iteración. El *SOST* está compuesto por varios módulos para un total de 173, donde su mayoría no utilizan el modelo-vista-controlador en este caso se estudió delicadamente la codificación de modo tal que si sacamos el fragmento de cogido que sería el controlador de nuestro (M.V.C) no dañemos el funcionamiento del módulo. Según el análisis realizado se llegó a una conclusión de definir y aclarecer cual va hacer nuestro modelo, nuestra vista y nuestro controlador.

- Modelo: En cada módulo se tiene un archivo llamado index_sql. Php el cual es el modelo de datos donde se llevan a cabo todas las operaciones y consultas a la base de datos *MySQL* para la finalidad del módulo.
- Vista: La estructura visual que los operadores del software emplean en el día a día, se encuentra en el archivo index.Php donde se realizó los cambios pertinentes para la exclusión de funciones, clases y código css reestructurando el archivo.
- Controlador: Cuando se empezó el proyecto en su mayoría de módulos no se maneja el patrón (M.V.C) por tal razón se realizó la exclusión de las funciones del index.Php donde cumplen el papel del controlador del módulo ya que maneja el intercambio de datos y la validación para el buen funcionamiento el nuevo archivo se llamó funciones.js

4.2.3 Ejecución de la iteración

4.2.3.1 Exclusión de funciones JavaScript. Al interactuar con *SOST* se encontró con una variedad de cantidad de archivos, teniendo en cuenta que la mayoría de módulos repiten su estructura y su interfaz gráfica pero su funcionalidad es independiente uno de otros. En su mayoría los módulos estaban compuestos como se ilustra en la ilustración 2 con archivos index.Php y index_sql.Php donde la idea es adaptar el (M.V.C) acomodando de manera eficiente y concreta el respectivo código de cada módulo.

ILUSTRACIÓN 8. REESTRUCTURACIÓN INDEX.PHP

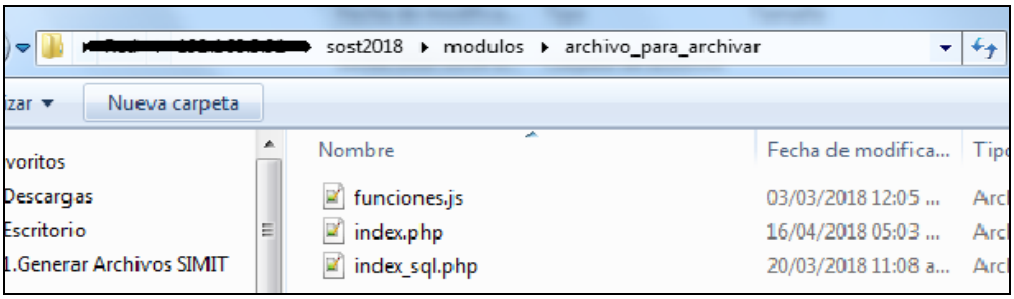



```
<?php
include_once ("../../js/valida_actividad.php");
require_once("../../config_s/config.php");
require_once("../../config_s/sql.php");
include_once ("../../js/calendario.php");

function principal($aid){
    $randos=rand();
?>
    <link rel="stylesheet" href="../../alertifyjs/css/themes/bootstrap.css?<?php echo $randos; ?>" />
    <link rel="stylesheet" href="../../alertifyjs/css/alertify.css?<?php echo $randos; ?>" id="toggleCSS" />
    <script src="../../alertifyjs/alertify.min.js?<?php echo $randos; ?>"></script>
    <script src="../../assets/js/jquery.2.1.1.min.js"></script>
    <link rel="StyleSheet" href="../../themes/style/style.css?<?php echo $randos; ?>" type="text/css">

    <link rel="stylesheet" href="../../css/tab-view.css?<?php echo $randos; ?>" type="text/css" media="screen">
    <script type="text/javascript" src="../../js/ajax.js?<?php echo $randos; ?>"></script>
    <script type="text/javascript" src="../../js/tab-viewi.js?<?php echo $randos; ?>"></script>

<script src="funciones.js?<?php echo $randos; ?>"></script>
<body>
<div align="center">
<form action="" name="formu_rep" id="fondoform">
<div id="contenedor" align="center">
        <div id="titulo_principal" >
        <a style="text-decoration:none; color:black; cursor: pointer" href="" onClick="recargar();">reportes inspeccion</a></div>
        <div>Seleccione los criterios necesarios.<br/>
            El tipo de reporte que desea ver:
            <select name="sel_reporte">
                <option value="audiencias" selected>Reporte Audiencias</option>
                <option value="estados">Reporte de Procesos</option>
                <option value="sansiones">Reporte de Sanciones</option>
                <option value="licencias">Reporte Licencias</option>
```

Como se puede observar en la ilustración 4 es la codificación antigua del index.Php de cada módulo, había una homogeneidad de distintos lenguajes de programación, comparándola con la ilustración 8 donde viene a ser la nueva estructura del index. Php. Ya que todos los módulos tienen dentro un archivo llamado index.Php este contiene el *JavaScript* donde es fundamental para el proceso del módulo, se debe saber que fragmento de código se puede excluir debido a que hay funciones donde su script tiene que estar ubicado en index.Php para su buen funcionamiento.

Uno de los principales cambios a realizar es transformar cada archivo que contiene la estructura de cada módulo en un modelo-vista-controlador, para ello se debe excluir y reubicar la codificación del lenguaje de programación *JavaScript* en un nuevo archivo donde va hacer el controlador de cada módulo.

ILUSTRACIÓN 9. NUEVO ARCHIVO JAVASCRIPT(FUNCIONES.JS)



Se puede evidenciar comparando con la ILUSTRACIÓN 2, que la creación de tal archivo era necesaria para poder adoptar el (M.V.C). Implementando esta operación en cada módulo activo de *SOST*, y permitir asegurar que se está manejando el (M.V.C), esta primera tarea es la más extensa, pues la cantidad de módulos, archivos y reportes son cuantiosos.

4.2.3.2 Adaptación del nuevo encabezado a los archivos index e index_sql.

Se deben modificar los index.php e index_sql.php de cada módulo, unificando las funciones y las variables necesarias para el inicio del funcionamiento de cada módulo. Para ello se reestructuro la codificación de cada encabezado, aprovechando que todos los módulos utilizaban las mismas funciones y las mismas variables se unificaron en un archivo y se colocaron las rutas respectivas para su inicialización.

La modificación se puede observar en ilustración 8 donde solo encontramos 4 líneas de encabezado y el resto de codificación es la estructura HTML que va hacer la vista que el operario manipulará, el encabezado simplificado que se puede observar.

ILUSTRACIÓN 10. ENCABEZADO SIMPLIFICADO DE CADA MODULO INDEX.PHP

```php
<?php
include_once ("../../js/valida_actividad.php");
require_once("../../config_s/config.php");
require_once("../../config_s/sql.php");
include_once ("../../js/calendario.php");

function principal($aid){
    $randos=rand();
?>
```

En el inicio de cada módulo modificado se pueden notar la inclusión de cuatro archivos que contienen las variables de sesión, validaciones, clases y funciones básicas para la inicialización de cada módulo, se utilizaron dos funciones include_once y require_once estas funciones permiten incluir los archivos que necesitemos para el buen funcionamiento, en este caso se incluyeron:

- **Valida_actividad.php:** Contiene el inicio de sesión, variables globales que utilizan los módulos para su funcionamiento y una función de un límite de tiempo de inactividad.
- **Calendario.php:** Es un script con la función de un calendario donde la mayoría de módulos utilizan la función calendario para llenar formularios.
- **Config.php:** Contiene la función de iniciar la conexión a la base de datos *MySQL*.
- **Sql.php:** Contiene las clases para poder realizar las consultas a la base de datos.

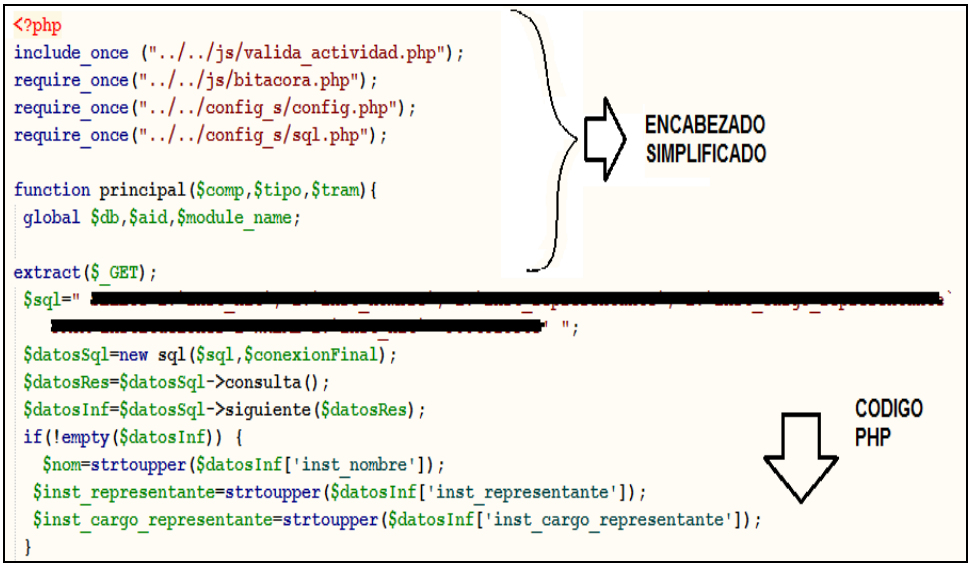
Como se puede comprender son archivos fundamentales para el correcto funcionamiento de cada módulo por ello son los que hacen parte del encabezado de cada módulo en el index.php.

Estas mismas líneas de código se ubicarán en los archivos de index_sql.php, que va a utilizar el módulo falta resaltar que cada módulo activo del software *SOST* debe contener he implementar el (M.V.C), el archivo index_sql.php utiliza las mismas rutas de los archivos mencionados anteriormente que se implementaban

en el index.php con una diferencia que se cambia el archivo de calendario por uno de bitácora más adelante se llevara a cabo la explicación de la nueva ruta implementada.

Los archivos index_sql.php también se modificó su encabezado donde los proporciona más seguridad y más fiabilidad en las rutas, se incluyeron archivos que son iguales a los de index.php con excepción del archivo calendario se agregó un archivo llamado bitácora esto se hizo con la intención de estar más seguros en los llamados de clases, conexión a base de datos e inicios de sesión.

ILUSTRACIÓN 11. COMPARACIÓN REESTRUCTURACIÓN INDEX_SQL.PHP



Se adiciono una herramienta la cual está en la inclusión llamada bitacora.php esta herramienta ya hacia parte del software pues ella llevará los datos de las acciones que se realicen en el software por los operadores, así se tendrá un control adecuado ya que permite llevar un registro escrito de diversas acciones. Su organización es cronológica, lo que facilita la revisión de los contenidos anotados.

Esta herramienta se adaptará a todos los módulos del software en el encabezado de index_sql.php, ya que es donde se encuentran nuestros modelo de cada módulo según el (M.V.C), vendría siendo la funcionalidad del módulo ya que

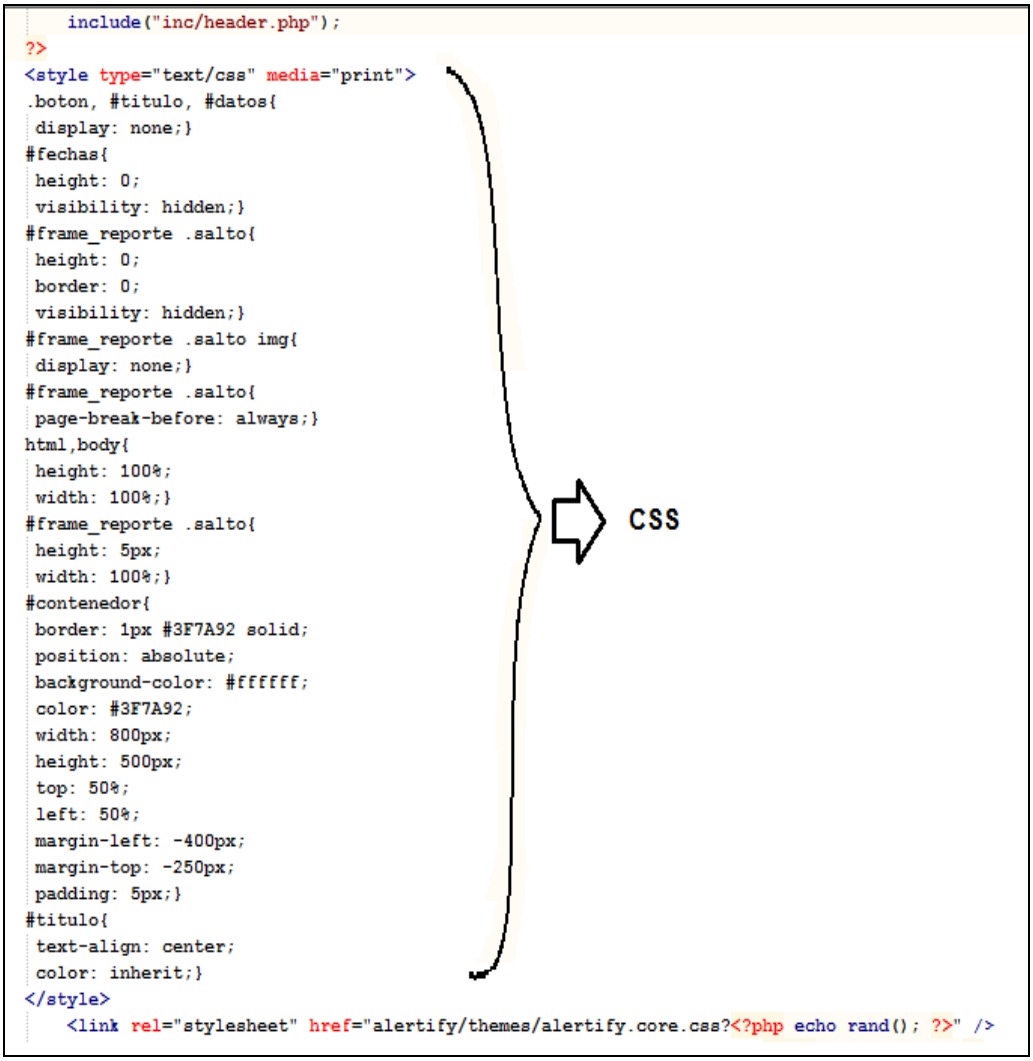
dependiendo de las consultas se realiza el desarrollo de la operación donde se guardará esa acción en una variable junto con el nombre el usuario u operador , la fecha y hora de la acción y desde que dirección Ip realizo la acción estos datos junto con otros son los administra la función bitácora.

ILUSTRACIÓN 12. BITÁCORA.PHP

```php
<?php
require_once("../../config_s/config.php");
require_once("../../config_s/sql.php");
$obj_conexion=new conexion();
$conexionFinal=$obj_conexion->abrir();

    if($proc==""){
    $bita_accion = $tipo;
    }else{
    $bita_accion = $proc;
    }

    $ultimo = strrpos($_SERVER['SCRIPT_FILENAME'], "/");
    $dl = substr($_SERVER['SCRIPT_FILENAME'], 12);
    $algo1 = $_SERVER['QUERY_STRING'];

    $sql_bitacora = "[redacted]";
    $q_bitacora = new sql($sql_bitacora,$conexionFinal);
    $q_bitacora->consulta();

    ?>
```

4.2.3.3 Exclusión de código css. Después de tener un encabezado limpio en cada módulo se dio a la tarea de excluir el código css donde su aporte en cada el módulo es dar un aspecto agradable, pues es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado, se creó una carpeta llamada *themes* está localizada afuera de todos los módulos donde se guardaran todos los estilos css a utilizar en el *SOST*, por medio de una ruta se llama los estilos requeridos para cada módulo ya sea por medio de clases, o etiquetas HTML.

ILUSTRACIÓN 13. CÓDIGO CSS DE CADA MODULO INDEX.PHP



Uno de los objetivos es dejar el software con el código puro posible sin necesidad de implementar plugin o framework ya sean para hacer funciones o aplicar diseños al software

La codificación *CSS* es una herramienta muy poderosa que con ayuda de *HTML* y *JavaScript* es muy usado para establecer el diseño visual de los documentos web, e interfaces de usuario escritas en *HTML o XHTML;* en este caso se utilizó solo *CSS* en los módulos a actualizar y se anexaron dos plugin o framework que se implementara en la interfaz general y en el sistema de *Alert*as.

ILUSTRACIÓN 14. EXTRACCIÓN DE CÓDIGO CSS DE CADA MODULO INDEX.PHP

```
<?php
include_once ("../../js/valida_actividad.php");
require_once("../../config_s/config.php");
require_once("../../config_s/sql.php");
include_once ("../../js/calendario.php");

function principal($aid){
        $randos=rand();
  $obj_conexion=new conexion();
    $conexionFinal=$obj_conexion->abrir();
?>
    <script src="../../assets/js/jquery.2.1.1.min.js"></script>
    <link rel="StyleSheet" href="../../themes/style/style.css?<?php echo $randos; ?>" type="text/css">
    <link rel="stylesheet" href="../../css/tab-view.css?<?php echo $randos; ?>" type="text/css" media="screen">
    <script type="text/javascript" src="../../js/ajax.js?<?php echo $randos; ?>"></script>
    <script type="text/javascript" src="../../js/tab-viewi.js?<?php echo $randos; ?>"></script>

<script src="funciones.js?<?php echo $randos; ?>"></script>
<body>
<div align="center">
<div id="contenedor">
<form action="" name="formu_rep">
 <td width="33%">
<div id="titulo_principal" ><a style="text-decoration:none; color:black; cursor: pointer" href="" onClick="recargar();">reportes general</a></div></td>
    <span id="datos">Seleccione los criterios necesarios.<br/>
        El tipo de reporte que desea ver:
     <select name="sel_reporte">
<option value="recaudo_neto">          Recaudo neto de transito       </option>
<option value="recaudo_concep">        Recaudo neto de transito/concep     </option>
<option value="recaudo_pago_externo">  Recaudo Pagos por Consignación      </option>
<option value="recaudo_syc">           Recaudo SYC CLT                     </option>
<option value="transferencias_simit">  Transferencias Simit    </option>
<option value="r_gruas">               Ingresos Gruas          </option>
<option value="e_alcaldia">            Estampillas Alcaldia    </option>
<option value="detalle_estampilla">    Detalle Estampillas Alcaldia - WS   </option>
          </select>
  </span>
  <br/>
```

RUTAS DE ARCHIVOS CSS

Como se observa se pueden comparar la codificación unificando el lenguaje de programación CSS, el cual nos aporta el estilo a nuestras platillas y formularios del *SOST*. El proceso realizado en este bloque fue extraer el código css, como se observa el cambio en ilustración 14, el cual se puede comparar con la ilustración 13 donde es el código original de *SOST*, se denota que el contenido del lenguaje de programación css, es muy extenso y hace que nuestro index tenga más líneas de contenido por lo que significa que cada módulo va a pesar más y por ende no hace más ágil la ejecución código.

Con la unificación de estilos css en un solo archivo y llamándolos por medio de rutas y requiriendo dicho estilo necesario para el desarrollo del módulo por medio de clases o etiquetas HTML va hacer más eficiente la actualización de los estilos y

a su vez su ejecución será más rápida ya que la carga del archivo css en el navegador se hará junto con la carga del archivo HTML y JS.

4.2.3.4 Exclusión de llamado Ajax. Como se mencionó en las herramientas utilizadas en el proyecto una de ella es una técnica de desarrollo web para crear aplicaciones interactivas, esta herramienta se unifico ya que cada módulo la contenida dentro de sus funciones de *JavaScript* esto producía problemas al abrir el software en otro navegador diferente a Mozilla Firefox ya que los llamado *Ajax* no todos estaban adaptados para diferentes navegadores.

ILUSTRACIÓN 15. LLAMADO *AJAX* 1 QUE SE UTILIZABAN.

```
<script type="text/javascript" src="js/alertas.js?<?php echo $randos; ?>"></script>
<body >

<script type="text/JavaScript" language='javascript'>

if (window.XMLHttpRequest){
   xmlhttp=new XMLHttpRequest();
        xmlhttp_total=new XMLHttpRequest();
}else{
   xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    xmlhttp_total=new ActiveXObject("Microsoft.XMLHTTP");
}
```

FUNCIÓN AJAX

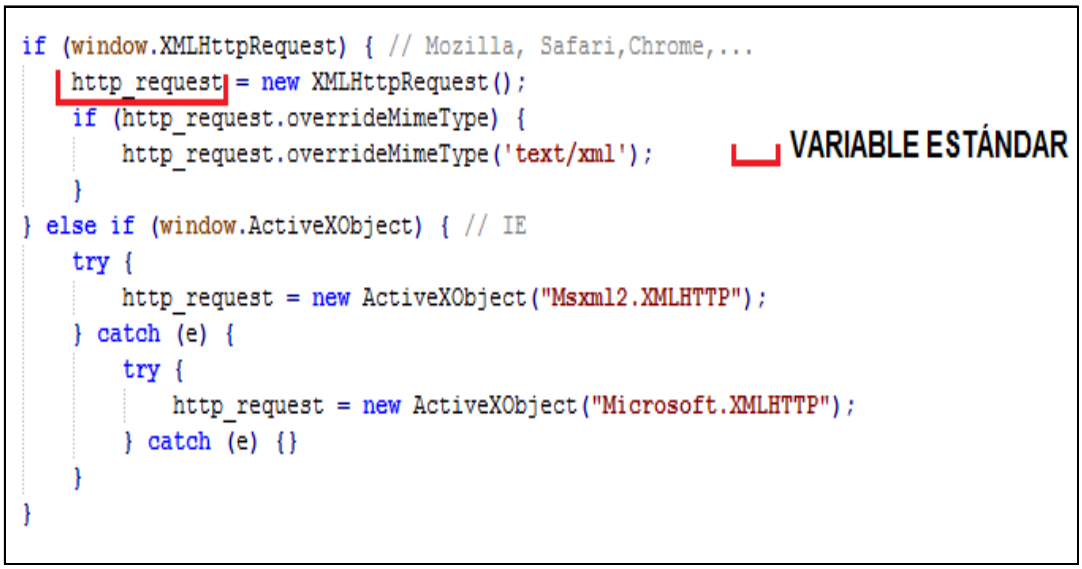
ILUSTRACIÓN 16. LLAMADO *AJAX* 2 QUE SE UTILIZABAN.

```
}
// Inicializamos AJAX
function inicia_AJAX(){
    var req = false;
    try
    {
    req = new XMLHttpRequest();
    }catch(err1){
        try
        {
            req = new ActiveXObject("Msxml2.XMLHTTP");
        }catch(err2){
            try
            {
                req = new ActiveXObject("Microsoft.XMLHTTP");
            }catch(err3){
                req = false;
            }
        }
    }
    return req;
}
function sel_indicativo(valor,tel){
    xmlhttp.onreadystatechange=function()
    {
        if(xmlhttp.readyState==4 && xmlhttp.status==200)
        {
            var telefono=document.getElementsByName(tel)[0];
```

FUNCION AJAX

Como se observa en las imágenes ilustración 15 e ilustración 16 son llamados *Ajax* diferentes de cada módulo, ya que su estructura *Ajax* son desiguales esto se debe a que se adecuó a el respectivo modulo que lo implementa, de ahí se obtenía la problemática de abrir *SOST* en navegadores distintos, ya que hay unas inicializaciones que no permiten el desarrollo de algunos módulos en diferentes navegadores, se llegó a una medida a los operadores de solo utilizar el *SOST* con el navegador Mozilla Firefox.

ILUSTRACIÓN 17. LLAMADOS *AJAX* UNIFICADO.



Se retiro los llamados *Ajax* de cada módulo y se remplazó la variable que iniciaba el proceso del envió de datos por una variable estándar para todos los módulos ya que la función de *Ajax* se ubicó afuera de los módulos, como las conexiones de base de datos y variables de inicio de sesión, se hizo la respectiva inclusión en el encabezado de cada módulo, haciendo más fácil la creación de módulos no tiene que crear la función del llamado *Ajax* solo utiliza la variable estándar al momento de necesitarlo.

4.2.3.5 Adaptación general del nuevo archivo funsiones.js. El software *SOST* en su mayoría sus módulos no contenían el patrón (M.V.C), como se ha venido desarrollando, se extrajo código del index.Php el cual fue almacenado en un nuevo archivo llamado funsiones.js este nuevo archivo solo contiene código de lenguaje *JavaScript* en que se desarrollan las validaciones e intercambio de datos a través del llamado *Ajax*, dependiendo la funcionalidad del módulo.

En este archivo se realizaron los ajustes pertinentes para su óptimo funcionamiento, convirtiéndose en uno de los tres pilares importantes para conformar el (M.V.C).

ILUSTRACIÓN 18. ARCHIVO *JAVASCRIPT* ANTIGUO.

```
    }
    rowstart=ultimo;
  }
  xmlhttp.onreadystatechange=function()
  {
    if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
        document.getElementById("InfoSost").innerHTML=xmlhttp.responseText;
        document.bancos.rowstart.value=rowstart;
    }
  }
  xmlhttp.open("GET","modules/bancos/index_sql.php?ban=Mysql&proc=Listado&frm_cod=41&rowstart="+rowstart,true);
  xmlhttp.send();
}

/*Inicia proceso de guardado en la tabla */
function GuardarConsulta(datos)
{
  xmlhttp.onreadystatechange=function()
  {
    if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
      alert(xmlhttp.responseText);
      Cerrar(document.bancos);
    }
  }
  var codi=document.bancos.codi.value;
  var nomb=document.bancos.nomb.value;
  xmlhttp.open("GET","modules/bancos/index_sql.php?ban=Mysql&proc=GuardarConsulta&frm_cod=41&codi="+codi+"&nomb="+nomb,true);
  xmlhttp.send();
}
```

RUTAS ANTIGUAS

La imagen es un pequeño fragmento del código JavaScript, ya que manejan todo lo que controla el módulo las funciones, las validaciones y él envió de los datos mediante *Ajax*. La magnitud del archivo funciones.js, dependen de que tan complejo es el módulo y su funcionamiento por ello al estar en el mismo archivo del index.php, se generaba gran cantidad de líneas de código en un solo archivo y hacia más complicado hacer un mantenimiento, actualizar una función, o corregir una validación.

Por ello sus rutas eran adaptadas según su estructura, también se puede observar las variables antiguas de la función *Ajax*, y al adaptar la refactorización del *SOST* se deben modificar las rutas de envió de variables.

ILUSTRACIÓN 19. ARCHIVO *JAVASCRIPT* MODIFICADO.

```
  }
  http_request.onreadystatechange=function()
  {
    if (http_request.readyState==4 && http_request.status==200)
    {
        document.getElementById("InfoSost").innerHTML=http_request.responseText;
        document.bancos.rowstart.value=rowstart;
    }
  }
  http_request.open("GET","index_sql.php?ban=Mysql&proc=Listado&frm_cod=41&rowstart="+rowstart,true);
  http_request.send();
}

/*Inicia proceso de guardado en la tabla */
function GuardarConsulta(datos)
{
  http_request.onreadystatechange=function()
  {
    if (http_request.readyState==4 && http_request.status==200)
    {
      alertify.success(http_request.responseText);
      Cerrar(document.bancos);
    }
  }
  var codi=document.bancos.codi.value;
  var nomb=document.bancos.nomb.value;
  http_request.open("GET","index_sql.php?ban=Mysql&proc=GuardarConsulta&frm_cod=41&codi="+codi+"&nomb="+nomb,true);
  http_request.send();
}

/*Inicia proceso de editar la información en la tabla */
function EditarConsulta(datos)
{
  http_request.onreadystatechange=function()
  {
```

VARIABLE ESTANDAR AJAX

NUEVA RUTA (M.V.C)

Al tener el nuevo archivo se tienen que hacer varios cambios, pero el de mayor importancia es ubicar bien las rutas ya que es un nuevo archivo las rutas cambian

y se deben redireccionar, también se realizaron varios cambios en este archivo los cuales ya fueron mencionado anteriormente.

4.2.4 Inspección y adaptación. Al completar el primer bloque de tareas se llevó a cabo una reunión para analizar los cambios realizados, se realizaron pruebas para observar la funcionalidad del módulo y que este siga en su buen funcionamiento, el ingeniero que está a cargo del proyecto aprobó los cambios realizados, no encontró ninguna eventualidad

4.3 TAREA 3: ADAPTACIÓN DE INTERFAZ GENERAL Y PLUGIN DE *ALERT*AS (INFORMATIVAS).

4.3.1 Adaptación. Entramos a una tercera etapa del proyecto donde según la metodología scrum es nuestro tercer bloque de tareas a realizar, como el proyecto consiste en actualizar las funciones y clases a su vez adaptando la estructura a un modelo-vista-controlador se aprovechó la manipulación del código interno para cambiar un poco la parte estética de los módulos, se deja claro que al cambiar lo estético la funcionalidad de los módulos no va cambiar y siguen siendo los mismos procesos o pasos que los usuarios u operadores deben realizar para el funcionamiento del módulo.

4.3.2 Planificación de la iteración. El ingeniero a cargo del proyecto por parte de la empresa ya tenía una interfaz general a implementar y unas *Alert*as en el software *SOST*, donde evaluó la utilización de los *Framework* a utilizar, ya que hay que tener cuidado con las licencias pues es un tema delicado jurídicamente, ya que se debe tener la completa seguridad que no va suceder nada en contra de la empresa.

4.3.3 Ejecución de la iteración.

4.3.3.1 Implementación de nueva interfaz gráfica general.

- **Ingreso o *Login*:**

Como es de saberse el ingreso o login es un proceso mediante el cual un usuario accede a una cuenta informática, sitio web, o un software, este tipo de proceso

suele ir acompañado primero de un previo registro y segundo por el ingreso de un ID de usuario y una contraseña *o password*.

En este proceso estaba complementado por una interfaz gráfica donde se implementó solo código css, unas validaciones y funciones consolidadas por *JavaScript* las cuales se valida el usuario y la contraseña correcta, la función de recordar la contraseña e invalidar sesión.

ILUSTRACIÓN 20. INGRESO O LOGIN ANTIGUO DE *SOST*.

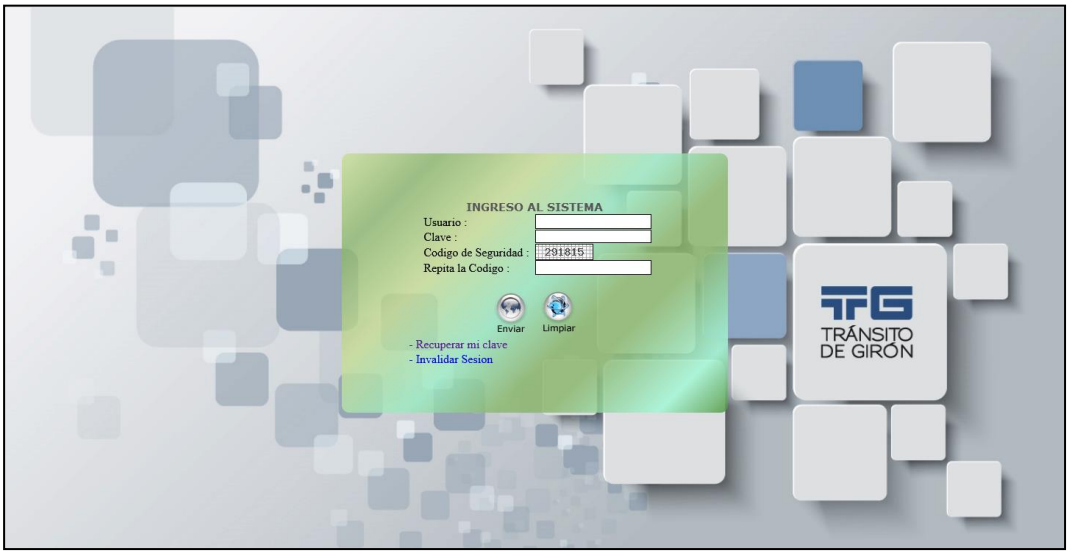
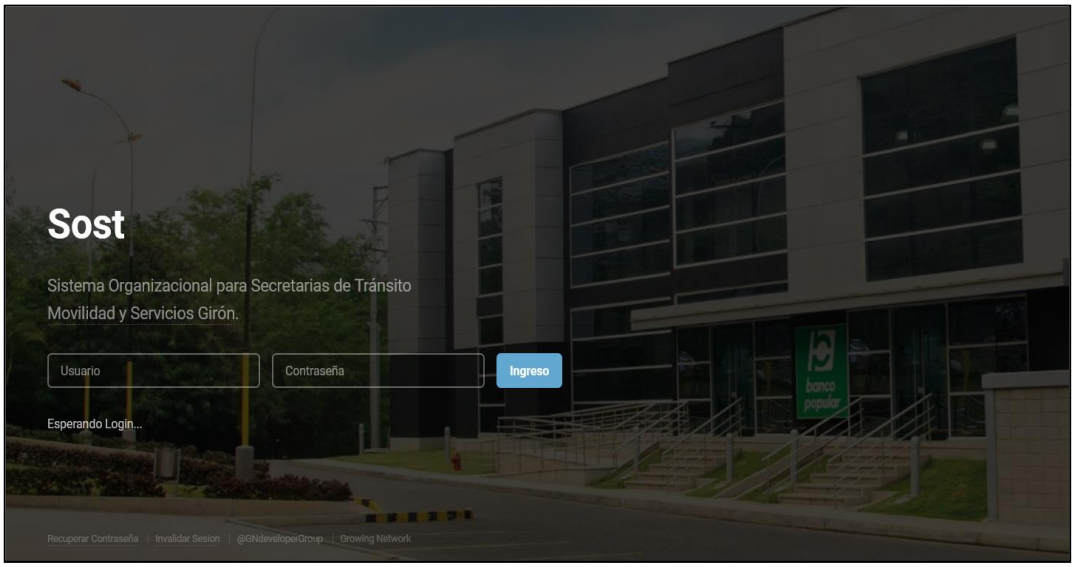


ILUSTRACIÓN 21. INGRESO O LOGIN MODIFICADO DE *SOST*.



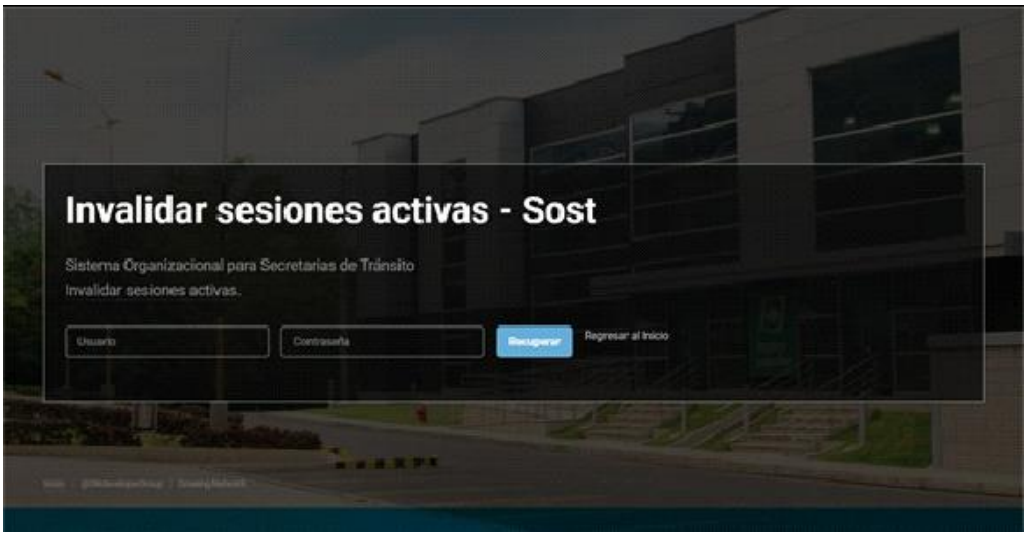
A simple vista se ve un cambio radical pues los colores, las imágenes son más opacas y no son tan coloridas como en el anterior *login*, además se adaptaron fotografías de la empresa, los efecto que se utilizaron son muy sencillos pues se adaptó la fotografía se colocó un color oscuro con transparencia, también se adaptaron dos cuadros de texto, que a comparación del anterior login se tenía una tabla con tres cuadros de texto y una imagen el cual era la función del captcha el cual consiste en que el usuario introduzca correctamente un conjunto de caracteres que se muestran en una imagen distorsionada que aparece en pantalla. Se supone que una máquina no es capaz de comprender e introducir la secuencia de forma correcta, por lo que solamente el humano podría hacerlo,
se eliminó esa función en la nueva entrada o *login*, al implementar el *SOST* en la empresa se, va hacer la entrada desde una dirección especial la cual cada usuario u operador del *SOST* tiene asignada una *Ip* única para el ingreso al sistema.
Una de las dos opciones que ofrece el login es la recuperación de la contraseña ya que el *SOST*, maneja un protocolo de seguridad, que por mínimo cada dos meses se debe cambiar la contraseña, debido que los operadores manejan variedad de contraseñas en el trascurso del tiempo se han presentado casos donde olvidan la contraseña.

ILUSTRACIÓN 22. RECUPERACIÓN DE CONTRASEÑA.

Debe ingresar una contraseña antigua que alguna vez utilizo, esta contraseña una vez utilizada no volverá a servir para recuperar otra contraseña queda inservible, después se ingresa la nueva contraseña la cual queda por defecto para el ingreso al sistema.

Invalidar sesión es la otra actividad que ofrece el *login*, ya que al ingresar al sistema hay una validación especial que impide el ingreso al usurario teniendo otra cuenta abierta, es decir *SOST* no permite el ingreso de dos sesiones en diferente computador de un mismo usuario, por ello esta este módulo que permite cerrar sesión en el computador donde se estaba trabajando antes, para poder ingresar en otro computador distinto.

ILUSTRACIÓN 23. INVALIDAR SESIONES ACTIVAS.



- **Menú:**

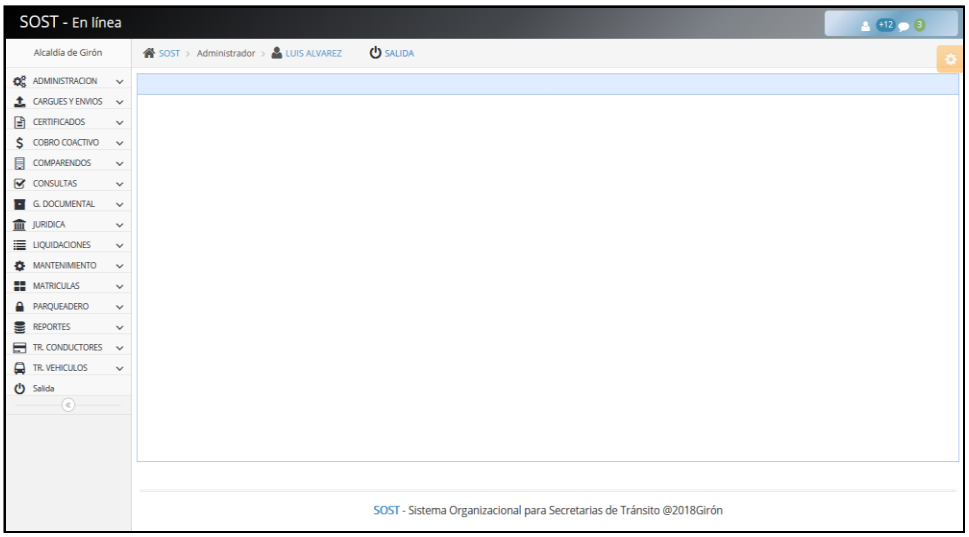
Se aprovechó la manipulación y la reestructuración interna que se realizó al implementar el modelo-vista-controlador para cambiar un poco lo estético del software por medio de un template o plugin que el ingeniero a cargo ya tenía predestinado. Este diseño está compuesto de dos partes el login de entrada y el menú el login no se adaptó el que trae por defecto el plugin el login fue realizado por medio de *CSS* y el menú si le dio forma el template que traía por defecto. Se

van hacer las comparaciones respectivas de cómo va quedar *SOST* estéticamente.

ILUSTRACIÓN 24. MENÚ ANTES DE LA ACTUALIZACIÓN.

Como podemos observar este tema del menú no se actualizaba desde principios del 2016 ya que en el historial del *SOST* solo ha tenido dos actualizaciones gráficas con esta implementación sería la tercera en el tema de interfaz gráfica ya que el *SOST* diariamente se le hacen modificaciones para un mejor funcionamiento.

ILUSTRACIÓN 25. MENÚ DESPUÉS DE LA ACTUALIZACIÓN.



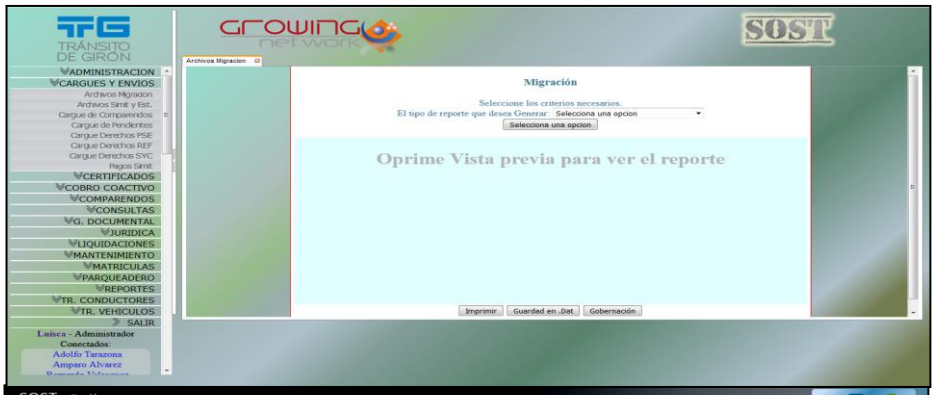
Es un template modificado pues se le adaptaron los colores según el login de entrada, para que no quedara tan disparejo este nuevo tema maneja unos colores más oscuros, donde no va hacer tan colorido como el anterior interfaz. Se empezó con el menú del software ya que es un template perfilado y adaptado para su uso.

- **Módulos:**

Se presentaran algunos de los 173 módulos que complementan el software *SOST*, ya que el sistema maneja muchos módulos que son iguales estéticamente pero la funcionalidad es completamente distinta, resaltando que la funcionalidad de cada módulo no va a cambiar, sigue siendo el mismo proceso de cada módulo. Este proyecto se trata de hacer una refactorización para soportar una actualización tecnológica y con ello se llevó a cabo la actualización de la interfaz gráfica no se manipulo ningún procedimiento para el buen funcionamiento de los módulos.

ILUSTRACIÓN 26. COMPARACIÓN DE INTERFAZ EN MÓDULOS DE SACAR REPORTES

- **ANTIGUA**



- **NUEVA**

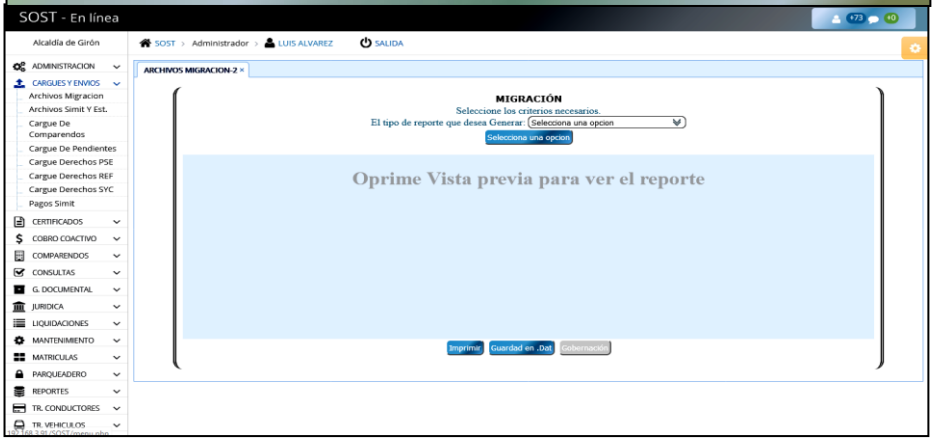


ILUSTRACIÓN 27. COMPARACIÓN DE INTERFAZ EN MÓDULOS DE ANULAR RECIBOS.

- **ANTIGUA**

- **NUEVA**

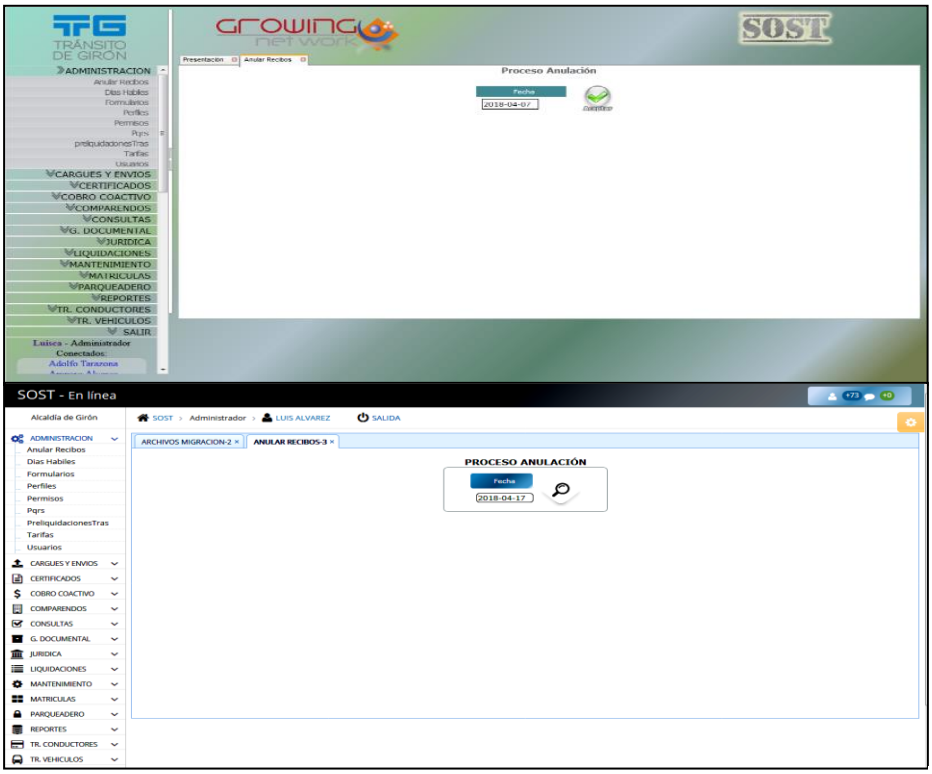


ILUSTRACIÓN 28. COMPARACIÓN DE INTERFAZ EN MÓDULOS DE TRAMITES

- **ANTIGUA**

- **NUEVA**

ILUSTRACIÓN 29. COMPARACIÓN DE INTERFAZ EN MÓDULOS DE CRUD.

- **ANTIGUA**

- **NUEVA**

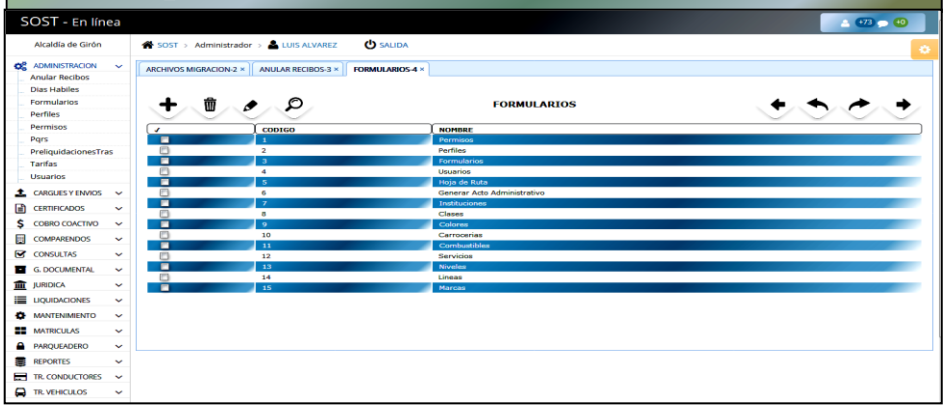
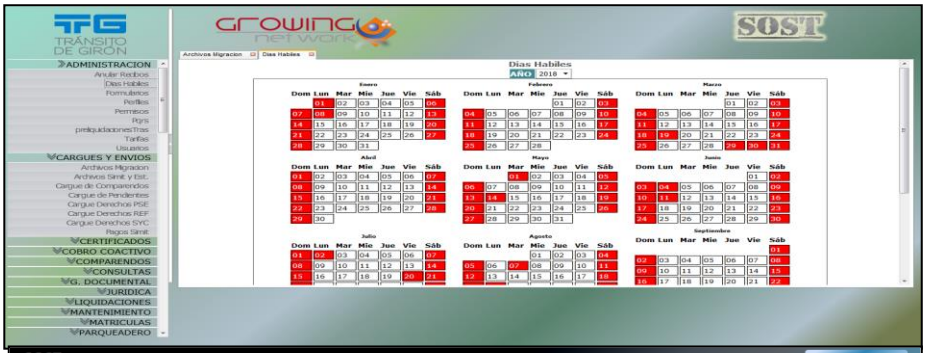


ILUSTRACIÓN 30. COMPARACIÓN DE INTERFAZ EN MÓDULO DE DÍAS HÁBILES.

- **ANTIGUA**



- **NUEVA**

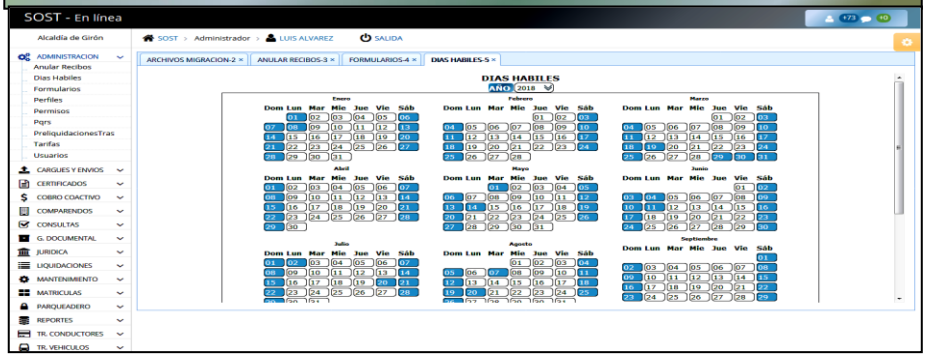
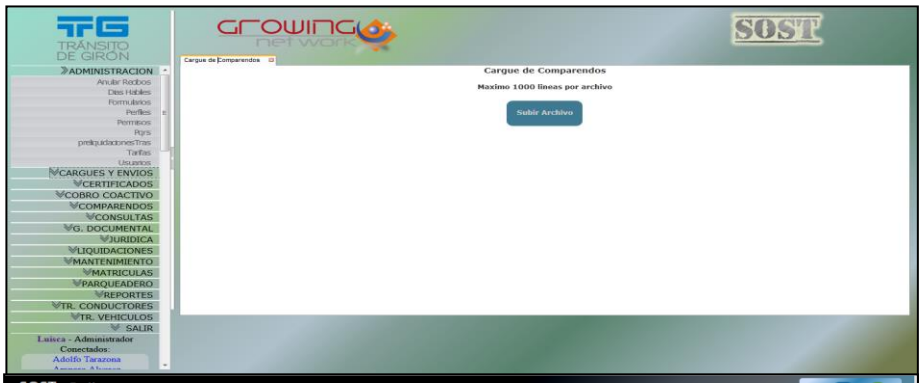


ILUSTRACIÓN 31. COMPARACIÓN DE INTERFAZ EN MÓDULOS DE INGRESAR REPORTES

- **ANTIGUA**

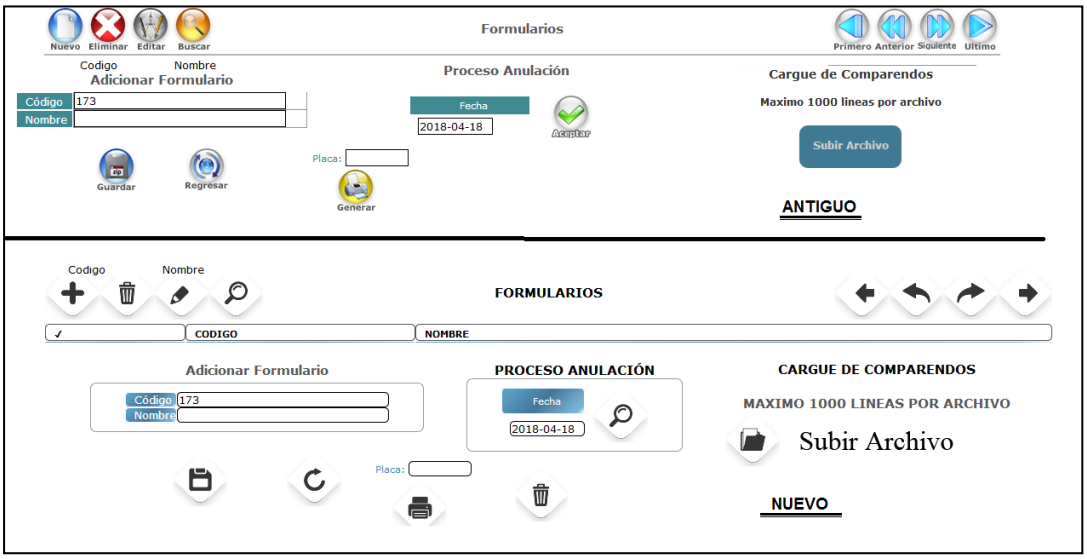


- **NUEVA**

Como podemos observar la estructura gráfica de los módulos antiguos es muy parecida al de la nueva interfaz, hay un cambio total en la parte de la botonera ya que manejaban variedades de colores y se le dio efecto por medio del lenguaje de programación css, como no lo muestra la ILUSTRACIÓN 25 una comparación de los bonotes nuevos con los antiguos.

El fuerte de este proyecto no estaba centrado en la interfaz gráfica, sino en la reestructuración del código para soportar una actualización tecnología, donde se aprovechó la manipulación del código fuente y se actualizo un poco la interfaz, también se hizo por comodidad a los que operan los módulos ya que tocaría invertir más tiempo en capacitaciones y la cantidad de gente que lo operan es considerable.

ILUSTRACIÓN 32. COMPARACIÓN DE LA BOTONERA EN LA INTERFAZ DE TODOS LOS MODULOS.



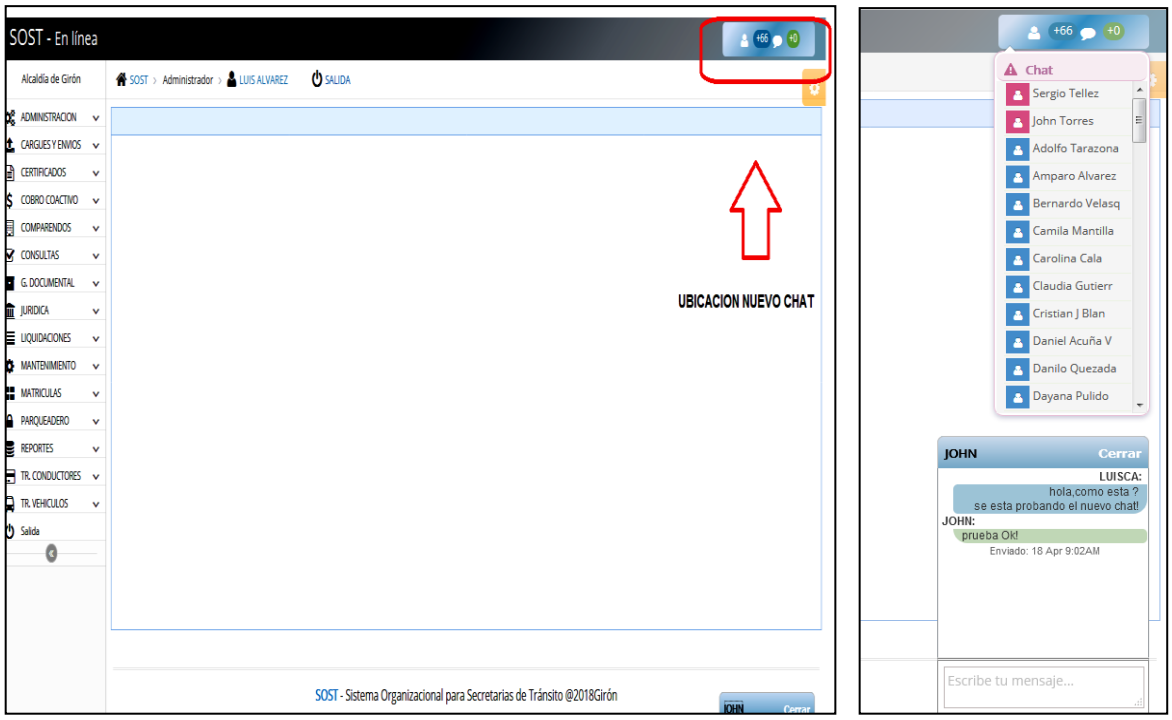
- **Chat interno del sistema *SOST*:**

El software maneja un sistema de comunicación en tiempo real que se realiza entre los usuarios que utilizan el sistema, esta aplicación ya la tenía implementada el sistema pues se hizo por la comodidad de la comunicación entre los que tienen acceso al sistema *SOST*, de no tener que estar desplazándose o comunicándose por líneas telefónicas teniendo la comodidad del chat.

ILUSTRACIÓN 33. CHAT EN LA INTERFAZ ANTIGUA.

Como podemos visualizar los usuarios conectados en la imagen izquierda son los que han ingresado al sistema, por ende tienen la sesión activa, y el sistema automáticamente los ubica en la lista y los visualiza para poder establecer una conversación si es necesaria, esta ubicación de los usuarios conectados está ubicada al final del submenú ubicado en el lado izquierdo, donde cada submenú contiene cierta cantidad de módulos dependiendo el departamento al que pertenece.

ILUSTRACIÓN 34. CHAT EN LA INTERFAZ NUEVA.



Esta modificación se hizo gracias a las opciones que proporcionaba el template que se aplicó, se cambió debido a mejorar una estética adecuada ya que se utilizaba el chat de una manera un poco desorganizada en la cuestión de los usuarios y las ventas de chat no tenían casi estilos css, como se observa se cambió la ubicación de los usuarios, también se les dio estilos a las ventas de chat , se le adiciono un funcionalidad que tiene que ver con los colores que se observan en los usuarios conectados parte superior derecha, el chat le da

prioridad a los chats con los que más se tienen actividad saldrán de primeros y de color rojo claro.

4.3.3.2 Implementación de plugin en *Alert*as informativas.

- ***ALERT* ():**

Como se ha venido documentando el software maneja una gran cantidad de validaciones en *JavaScript* donde muchos de los módulos utilizan validaciones que vendrían siendo *Alert*as, que indican si el proceso del módulo respectivo donde se esté trabajando esta quedan bien o mal, esta función la trae por defecto *JavaScript* se implementa de una manera muy sencilla solo escribiendo *Alert* ();

El método de *Alert* () muestra un cuadro de *Alert*a con un mensaje específico y un botón Aceptar. Con frecuencia se usa un cuadro de *Alert*a si desea asegurarse de que la información llegue al usuario.

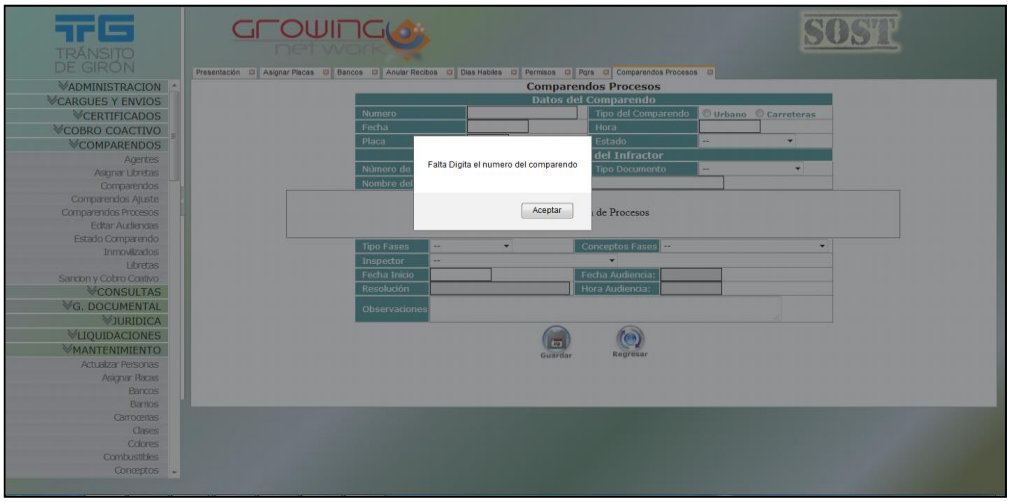
Dentro de los corchetes podemos colocar una variable donde podemos observar el valor que hay en ella o un string que es una cadena de caracteres, palabras, o frase; es una secuencia ordenada de elementos que pertenecen a un cierto lenguaje formal o alfabeto.

ILUSTRACIÓN 35. IMPLEMENTACIÓN DEL MÉTODO *ALERT*().

```
    <html>
<body>
<p>Es un ejmplo de como funciona</p>
<button onclick="mifuncion()"> Enviar</button>
    <script>
    var usuario = alexander;
    function mifuncion(){
        alert("hola,esto es una alerta")

        alert(usuario);
    }
    </script>
</body>
</html>
```

Es de una manera muy sencilla de implementarlo se puede colocar en cualquier lado dentro de un *script*, en este caso el software lo implementa para colocar los *Alert*as naturales en cualquier validación que la necesite

ILUSTRACIÓN 36. MÉTODO ORIGINAL *ALERT*() .



- ***CONFIRM*():**

Al realizar una confirmación de una búsqueda se utiliza el método que trae el lenguaje de programación *JavaScript* llamado *confirm*() ya que va acompañado de unas condiciones para hacer valido el método, El cuadro de confirmación quita el foco de la ventana actual y obliga al usuario u operador a leer el mensaje. No abuse de este método, ya que evita que el usuario acceda a otras partes de la página hasta que se cierre el cuadro.

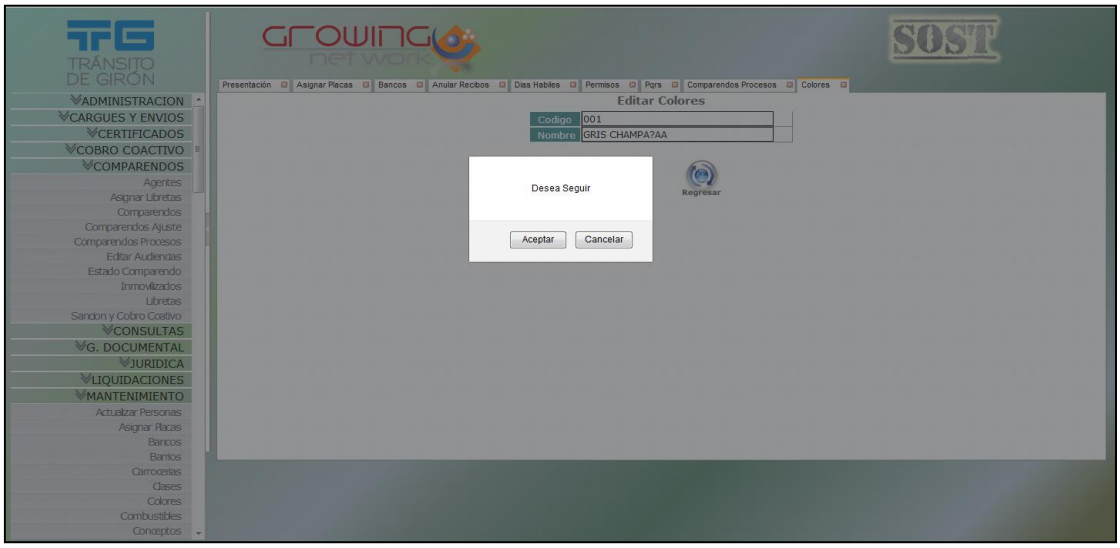
En el *SOST* se utiliza cuando se va hacer una búsqueda en algún modulo que tenga implementada la función, Con frecuencia, se utiliza un cuadro de confirmación si desea que el usuario verifique o acepte algo

ILUSTRACIÓN 37. IMPLEMENTACIÓN DEL MÉTODO CONFIRM ().

```
<!DOCTYPE html>
<html>
<body>

<p>Es un ejmplo de como funciona</p>

<button onclick="mifuncion()">Enviar</button>

<p id="demo"></p>

<script>
function myFunction() {
    var txt;
    var r = confirm("Desea seguir??");
    if (r == true) {
        txt = " OK!";
    } else {
        txt = "Cancelar!";
    }
    document.getElementById("demo").innerHTML = txt;
}
</script>

</body>
</html>
```

Estos dos métodos que se acabaron de enseñar se implementan en casi todos los módulos por ello se decidió darles un aspecto más agradable y no mostrar el que tiene el lenguaje de programación por defecto.

ILUSTRACIÓN 38. MÉTODO ORIGINAL CONFIRM() .



- ***PROMPT* ():**

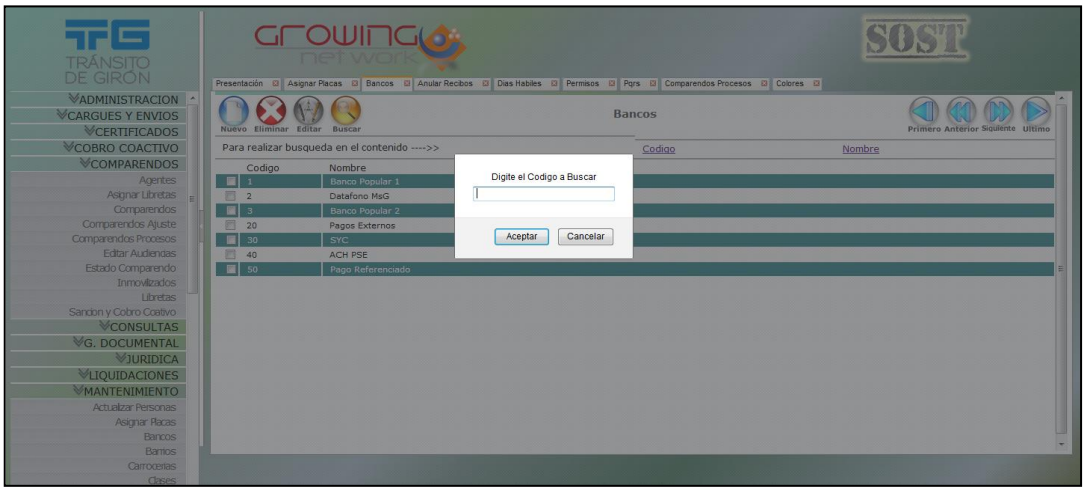
Estas dos *Alert*as presentadas solo tienen una única interacción donde solo, es dar clic en sus opciones, pero también se utiliza una ventana emergente que tiene parentesco a un *Alert* el método es window.*Prompt*();

ILUSTRACIÓN 39. IMPLEMENTACIÓN DEL METODO *PROMPT*.

```
//promt
aquello=window.prompt('Digite el texto a Buscar','');
if(aquello)  {
    //busqueda
  }else{
    //busqueda cancelada
  }
```

El método *Prompt* () muestra un cuadro de diálogo que solicita la entrada del visitante. A menudo se usa un cuadro de aviso si desea que el usuario ingrese un valor antes de ingresar a una página, en el *SOST* se utiliza a la hora de utilizar la función buscar, esta función esta disponibles en los módulos cuyo contenido es un Crud de contenido, ahí podemos buscar información por medio de este método *Prompt* () el cual llama una función de buscar

ILUSTRACIÓN 40. MÉTODO ORIGINAL *PROMPT*() .



La nueva adaptación que se realizó para el sistema de *Alert*as fue un plugin cuyo efecto es muy dinámico y hace que el módulo tenga interacción con el usuario, no

un letrero básico sin colores, ni estilos css, que indique que está bien y que está mal. El plugin que se adaptó al software se llama *ALERTIFYJS* donde es un *framework* de *JavaScript* para desarrollar elegantes diálogos y notificaciones del navegador, esta herramienta es muy buena para darle una estética adecuada a las validaciones de cada módulo y salir de los estilos que trae por defecto los lenguajes de programación en el tema de las *Alert*as, se descargó el *framework* se adaptó al software y la manera de poder implementarlo no se podía asignar el nueva *Alert*a de manera masiva debido a que manejamos tres tipos de *Alert*as distintas, la actualización se hizo una por una en cada módulo lo cual consumió un buen tiempo en implementarla todos los módulos.

- ***ALERT* () MODIFICADO:**

En este cambio se adiciono una *Alert*a ya que nos permitía informarnos de manera gráfica por medio de colores que proceso estaba bien o estaba mal, se manejaron dos colores, uno rojo que nos indica que el procedimiento está mal o el valor ingresado en el módulo es erróneo, además de eso si no se arregla el problema o no se llena los datos que requiere el módulo, el letrero no va a dejar de salir también, se maneja
el color verde donde informa que el proceso realizado fue exitoso. Hay que resaltar que este tipo de *Alert*a no bloquea la pantalla como lo hacía el nativo de *JavaScript*, el con el trascurso de 8 segundos el tiempo suficiente para leer el mensaje se desaparece automáticamente, se puede oprimir las veces que quiera él siempre va a salir hasta corregir el error.

ILUSTRACIÓN 41. CODIFICACIÓN DE *ALERTIFY* DE NEGACIÓN.

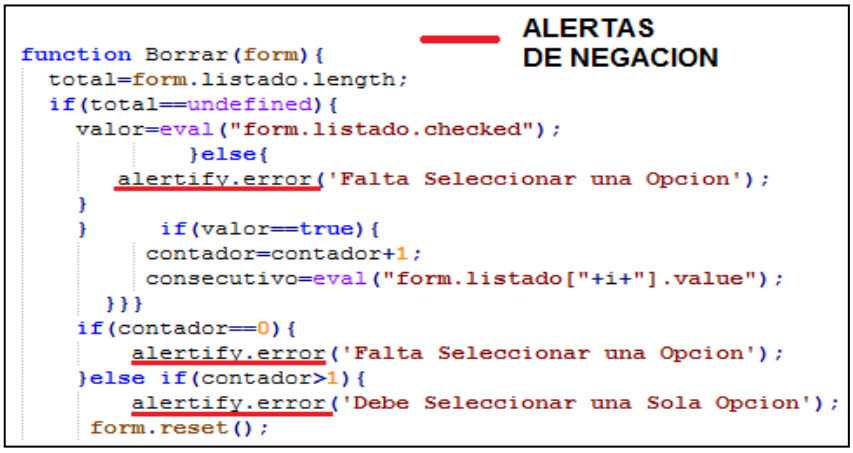
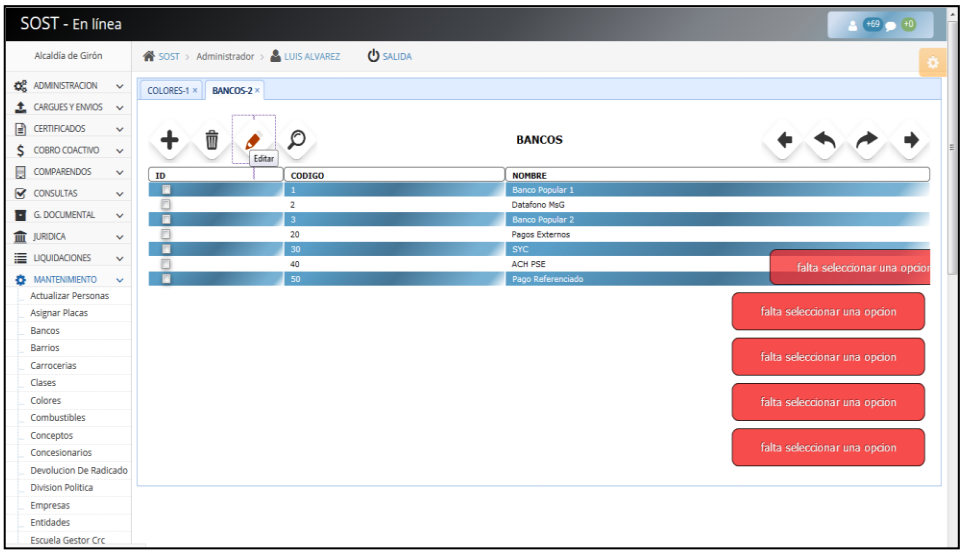


ILUSTRACIÓN 42*. ALERTIFY* DE NEGACIÓN IMPLEMENTADO.



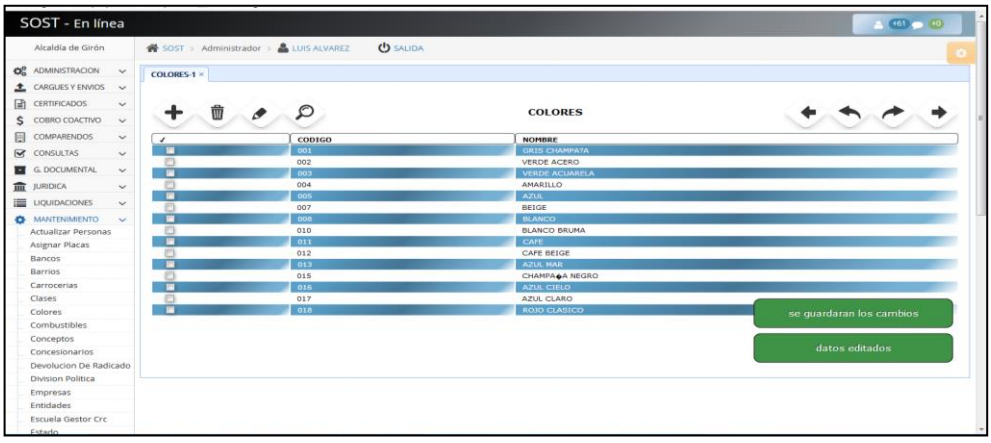
Como se puede evidenciar en la ILUSTRACIÓN anterior muestra la forma que aparecen las nuevas *Alert*ar negativas que indican errores y la manera de emplear el nuevo plugin, se debe incluir la librería en el encabezado de cada index de cada módulo y adaptar la estructura que requiere el plugin

ILUSTRACIÓN 43. CODIFICACIÓN DE *ALERTIFY* DE AFIRMACIÓN.

```
/*Inicia proceso de guardado en la tabla */
function GuardarConsulta(datos)
{
  http_request.onreadystatechange=function()
  {
    if (http_request.readyState==4 && http_request.status==200)
    {
      alertify.success(http_request.responseText);
      Cerrar(document.bancos);
    }
  }
  var codi=document.bancos.codi.value;
  var nomb=document.bancos.nomb.value;
  http_request.open("GET","index_sql.php?ban=Mysql&proc=GuardarConsulta&frm_cod=41&codi="+codi+"&nomb="+nomb,true);
  http_request.send();
}
```

ALERTAS DE AFIRMACION

ILUSTRACIÓN 44. *ALERTIFY* DE AFIRMACIÓN IMPLEMENTADO



Este tipo de *Alert*as informativas son de gran ayuda ya que por su color que la distingue se asegura que el proceso fue realizado con éxito, se adaptan iguales a las *Alert*as negativas.

- **CONFIRM () MODIFICADO:**

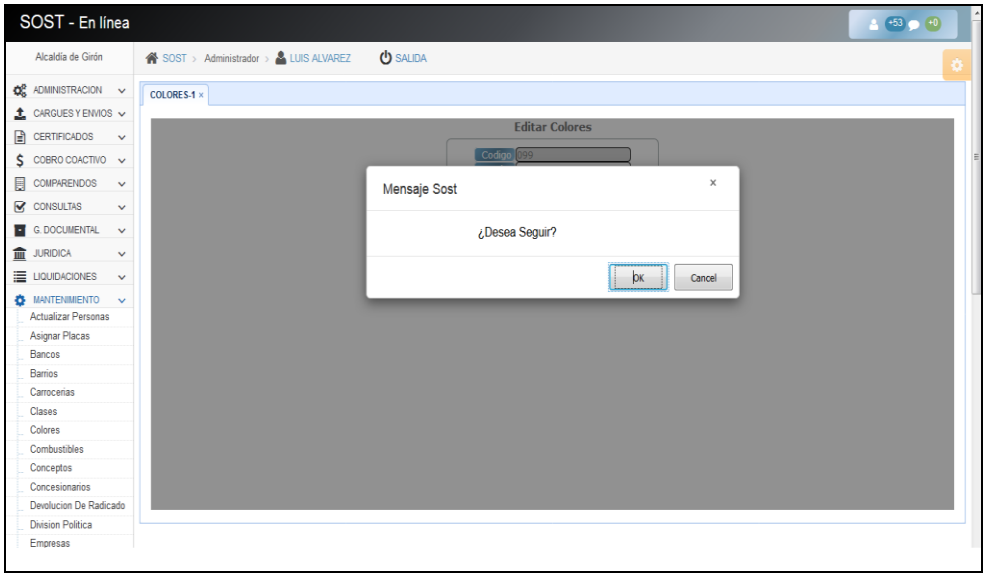
Este tipo de *Alert*a tiene un color neutro ya que el modulo no permite combinar variedad de colores, este tipo de *Alert*a informa sobre si deseamos concretar el proceso.

ILUSTRACIÓN 45. CODIFICACIÓN DE *ALERTIFY* DE CONFIRMACIÓN.

```
function Guardar(form)
{
 if(form.codi.value==''){
   alertify.error('Falta Digitar el Codigo');
   form.codi.focus();
 }else if (form.nomb.value==''){
   alertify.error('Falta Digitar el Nombre');
   form.nomb.focus();
 }else{
alertify.confirm('¿Desea Seguir?', function(){
    alertify.success("Se guardaran los cambios");
        if(form.proc.value=='Nuevo')
   {
     GuardarConsulta();
   }
   if(form.proc.value=='Editar')
   {
     EditarConsulta();
   }
```

ALERTA DE CONFIRMACIÓN

ILUSTRACIÓN 46. ALERTIFY DE CONFIRMACIÓN IMPLEMENTADO.



- ***PROMPT* () MODIFICACIÓN:**

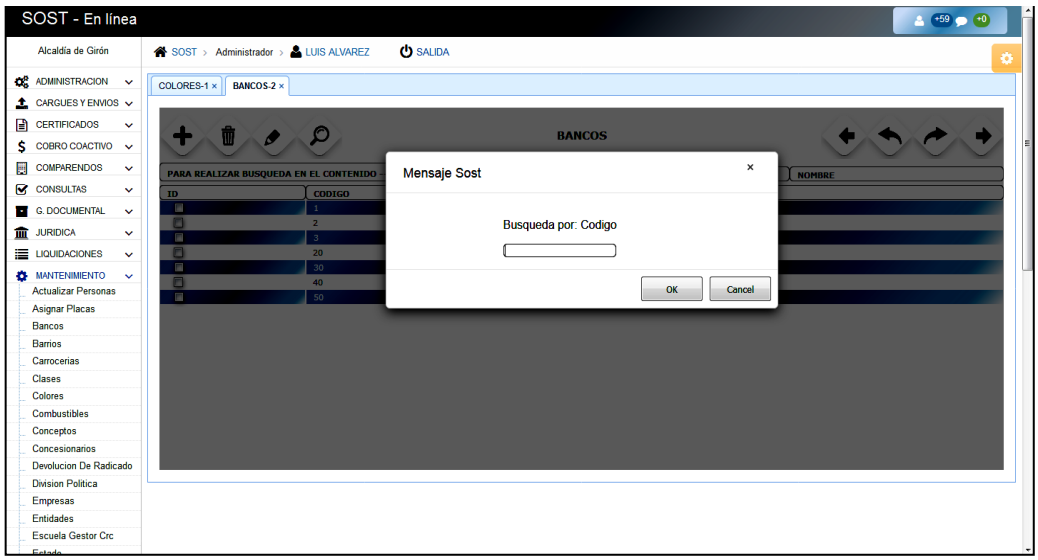
Este tipo de *Alert*a es de gran ayuda a la hora de hacer una búsqueda en *SOST* se implementó en todos los Crud (Crear, Leer, Actualizar y Borrar) para facilitar la búsqueda de datos ya que se maneja una gran cantidad de información en la mayoría de los Crud.

ILUSTRACIÓN 47. CODIFICACIÓN DE *ALERTIFY* DE SUGERENCIA.

```
/*Inicia proceso para enviar variables a la funcion listado*/
function Busqueda(form,nombre,caja){
    alertify.prompt( 'Busqueda por: ' + nombre, ''
                , function(evt, value) {
                http_request.onreadystatechange=function()
                {
                 if (http_request.readyState==4 && http_request.status==200)
                 {
                    document.getElementById("InfoSost").innerHTML=http_request.responseText;
                 }
                }
                http_request.open('GET','index_sql.php?ban=Mysql&proc=Listado&frm_cod=41&caja='+caja+'&aque='+value,true);
                http_request.send();
                }, function() {
            alertify.error('busqueda cancelada');
                });
```

ALERTA PROMPT

ILUSTRACIÓN 48. ALERTIFY DE *PROMPT()* IMPLEMENTADO.

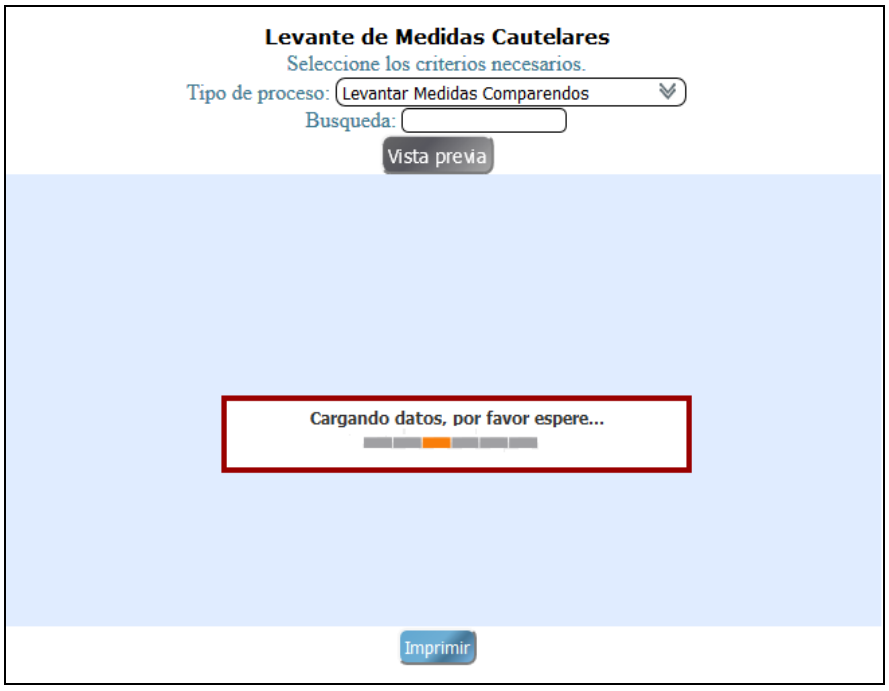


4.3.3.3 Implementación De *Loader*.

Al consultar un reporte, cargar un formulario o generar un dato en especial, el software hace una validación en el proceso de envió de datos el cual se hace por medio de *Ajax*, para poder traer la información requerida por el usuario u operario, en el trascurso de la operación se activa un *Loader* (Cargando…), el cual es un gif que indica que el archivo se está cargando, para demostrarle al usuario que el software está en un proceso.

Dicho gif es llamado por medio de una ruta, la cual lleva a un archivo por fuera de los módulos en general donde se encuentran todas las imágenes utilizadas por el *SOST*.

ILUSTRACIÓN 49. *LOADER* ANTIGUO.



Como se puede observar en la ilustración el gif no queda acorde con los colores de la nueva interfaz aplicada, además, es una imagen (gif) por lo general cada vez que se va a utilizar, el software tiene que ir a localizarlo y traerlo con la respectiva ruta. El nuevo *Loader* implementado ya no depende de una imagen (gif), ya que se adaptó por medio de estilos CSS, así no se insertan al software de tantas

imágenes y se gana más tiempo a la hora de llamarlo ya que los estilos CSS se cargan todos al cache al utilizar el software.

ILUSTRACIÓN 50. *LOADER* NUEVO.

4.3.4 Inspección y adaptación. Después de varios días que se implementaron en el desarrollo de este bloque de tareas se culminó con la tarea de manera rápida y efectiva. Se refiere a la manera rápida debido a que la manipulación fue modulo por modulo, cambiando tanto la interfaz interna de cada módulo como la adaptación de los nuevo *Alert*as donde se pudo avanzar de manera ágil. El ingeniero a cargo estuvo a favor del cambio, él corrigió algunos colores que no combinaban, esto se debe al cansancio que produce el contraste de los colores en los ojos, fue la única falencia en esta etapa la cual tuvo una solución muy rápida. Volviendo a resaltar que se trabajó directamente en el servidor, pero con una ruta distinta a la original que utiliza *SOST* en la empresa, también con una copia de una base de datos para no tener inconvenientes.

4.4 TAREA 4: ADAPTACIÓN DE CLASES EN LOS LLAMADOS *MySQL*.

4.4.1 Adaptación. Se puede denominar que este bloque de tareas es una de las más fundamentales para poder desarrollar la actualización del software a nivel de los lenguajes de programación; ya que por medio de esta trasformación de codificación se pueden actualizar el Php y el *MySQL,* lo cual es necesario poder implementar la orientación a objetos y eso se logra por medio de las clases, realizando esta tarea realizamos el objetivo fundamental en este proyecto.

4.4.2 Planificación de la iteración. Para poder desarrollar este bloque de tareas según la metodología, se tiene que adaptar un archivo especial donde se creara las clases a cambiar, en este caso solo se transformaran los llamados a la base de datos, todo lo que tenga que ver con la interacción con la base de datos, al tener un archivo especial solo con el contenido de las clases, se incluye en el index y en el index_sql una ruta especifica que habilite el contenido de dicho archivo, en estos dos archivos los cuales todos los módulos lo contienen debido a que ya se adaptó el Modelo-vista-controlador, tendremos que crear un objeto llamando las clases creada en tal archivo para sí poder implementarlas en cada consulta.

4.4.3 Ejecución de la iteración. Como primera medida se crearon las clases en una carpeta que está ubicada afuera de los archivos que contiene todos los módulos del *SOST*, esta organización se hace para unificar las clases y colocar una ruta especial y única para todos los archivos index y index_sql.

ILUSTRACIÓN 51. *ALERTIFY* DE CONFIRMACIÓN IMPLEMENTADO.

```php
<?php
include_once ("../../js/valida_actividad.php");
require_once("../../js/bitacora.php");
require_once("../../config_s/config.php");
require_once("../../config_s/sql.php");
```

Recordando el segundo bloque de tareas realizado se unificaron los archivos incorporando las rutas de archivos que me permiten el funcionamiento del módulo. Una de estas rutas lleva al archivo que contiene las clases; esa ruta es la que observamos dentro del recuadro de la imagen. Es claro que esta operación se tuvo que realizar de amanera incremental, uno por uno los archivos existentes de cada módulo.

***ILUSTRACIÓN 52*. ARCHIVO DE CLASES SQL.PHP.**

```php
<?php

$_SESSION["time"] = time();
class sql {
    var $sentencia;
    var $conexion;
    function sql($sql1, $conx1) {
        $this->sentencia = $sql1;
        $this->conexion  = $conx1;
    }
    function consulta() {
        //$resultado=mysql_query($this->sentencia,$this->conexion) or die (mysql_error());
        @$resultado       = mysql_query($this->sentencia, $this->conexion);
        return ($resultado);
    }
    function filas($resultado) {
        @$filas = mysql_num_rows($resultado);
        return ($filas);
    }
    function siguiente($resultado) {
        if (@$fila = mysql_fetch_array($resultado)) return $fila;
        else return NULL;
    }
    function verifica($resultado) {
        if ($resultado) $mensaje = "0";
        else $mensaje = "1";
        return ($mensaje);
    }
    function libera($resultado) {
        mysql_free_result($resultado);
    }
}
?>
```

La ruta conduce a la clase que se ve en la imagen, la actividad en este bloque es poder remplazar las funciones por la cases en cada módulo. Esto se hace de tal

manera cuando llegue el momento de actualizar los lenguajes de programación solo se tenga que cambiar este archivo y por medio de las clases se actualizan los 173 módulos del sistema *SOST*.

La modificación de cada módulo con lleva a un análisis de cuidado, y en especial el archivo index_sql como su nombre lo indica, están todas las consultas SQL y si se llega a manipular de mala manera algún dato en una consulta puede salir un reporte o un trámite con unos datos erróneos, lo cual sería un tema delicado para la empresa.

***ILUSTRACIÓN 53*. ARCHIVO INDEX_SQL.PHP SIN CLASES.**


```
<?php
if (!mb_eregi("archivo_enviar_archivo", $_SERVER['HTTP_REFERER']))
{
    die ("<script>location.href='../';</script>");
}

require_once("../../mainfile.php");
/*Realiza conexion con la base de datos*/
$obj_conexion=new conexion();
$conexionFinal=$obj_conexion->abrir();
extract($_REQUEST);
global $aid;
if(($_GET['ban']=="Mysql")&&($_GET['proc']=="pedir_placa")){
$sql_vehiculos="[redacted]";
$sql_q_vehiculo=mysql_query($sql_vehiculos, $conexionFinal);
$sql_c_vehiculo=mysql_num_rows($sql_q_vehiculo);
if($sql_c_vehiculo>0){
        $existe="Si";
    if($tpla_codigo=="placa"){
            $sql_cambio="[redacted] ";
        $sql_q_cambio=mysql_query($sql_cambio, $conexionFinal);
        $sql_c_cambio=mysql_num_rows($sql_q_cambio);
        if($sql_c_cambio>0){
            $existe="Si";
    }else{
        $existe="No";}}
        if($existe=="Si"){
$sql_entrega="[redacted]";
$sql_q_entrega=mysql_query($sql_entrega, $conexionFinal);
$sql_c_entrega=mysql_num_rows($sql_q_entrega);
    if($sql_c_entrega>0){
```

Esta imagen fue tachada en ciertas líneas de código que representan la integridad a los datos de la empresa y por seguridad fueron ocultas.

Esta imagen del archivo fue obtenida del módulo de archivo_enviar_carpeta.php el cual representa una función importante en la empresa pues son las que

administran las cantidades de placas requeridas ya que cada placa tienes su carpeta con los tramites y procesos que ha llevado la placa en el trascurso de su uso.

Las clases que se van a implementar van a sustituir de los módulos a las siguientes funciones

ILUSTRACIÓN 54. FUNCIONES A SUSTITUIR.

```php
<?php

/*Crear consulta*/
$datosSqlMarc=mysql_query($sql, $conexionFinal);

$datosSqlMarc1=new sql($sql,$conexionFinal) or die (mysql_error());
$datosSqlMarc=$datosSqlMarc1->consulta();


-----------------------------------------------------------------
/**cantidad de registros */
$info_cantidad=mysql_num_rows($datosSqlMarc);

$info_cantidad=$datosSqlMarc1->filas($datosSqlMarc);
-----------------------------------------------------------------
/*traer datos*/
$data=mysql_fetch_array($datosSqlMarc);

$data=$datosSqlMarc1->siguiente($datosSqlMarc);



?>
```

FUNCIONES NATIVAS

CLASES IMPLEMENTADAS

Se instanciaron las clases en cada archivo index_sql.php y se llama un objeto para unificar las funciones, así poder hacer una actualización del php de forma masiva, pero para poder realizarlo toca implementar modulo por modulo las clases y esta tarea llevara un buen tiempo ya que se manejan muchas consultas ya que se cuenta con una capacidad de 173 módulos.

Las funciones que se van a sustituir por las clases son tres:

- ***MYSQL*_QUERRY:** Enviar una consulta *MySQL*, envía una única consulta a la base de datos actualmente activa en el servidor asociado, Este

comando es únicamente válido para sentencias como SELECT, INSERT, UPDATE, REPLACE o DELETE.

- ***MYSQL*_NUM_ROWS:** Obtener el número de filas de un conjunto de resultados, Recupera el número de filas de un conjunto de resultados. Este comando es únicamente válido para sentencias como SELECT o SHOW que retornan un conjunto de resultados real
- ***MYSQL*_FETCH_ARRAY:** Recupera una fila de resultados como un array asociativo, un array numérico o como ambos.

ILUSTRACIÓN 55. ARCHIVO INDEX_SQL.PHP CON CLASES.

```php
<?php
include_once ("../../js/valida_actividad.php");
require_once("../../js/bitacora.php");
require_once("../../config_s/config.php");
require_once("../../config_s/sql.php");
/*Realiza conexion con la base de datos*/
$obj_conexion=new conexion();
$conexionFinal=$obj_conexion->abrir();
extract($_REQUEST);
global $aid;
if(($_GET['ban']=="Mysql")&&($_GET['proc']=="pedir_placa")){
$sql_vehiculos="[redacted]";
$obj_vehiculos=new sql($sql_vehiculos,$conexionFinal) or die (mysql_error());
$cons_vehiculos=$obj_vehiculos->consulta();
$sql_c_vehiculo=$obj_vehiculos->filas($cons_vehiculos);
if($sql_c_vehiculo>0){
    $existe="Si";
    if($tpla_codigo=="placa"){
        $sql_cambio="[redacted]";
        $obj_cambio=new sql($sql_cambio,$conexionFinal) or die (mysql_error());
        $cons_cambio=$obj_cambio->consulta();
        $sql_c_cambio=$obj_cambio->filas($cons_cambio);
        if($sql_c_cambio>0){
            $existe="Si";
    }else{
        $existe="No";}}

    if($existe=="Si"){
$sql_entrega="[redacted]";
$obj_entrega=new sql($sql_entrega,$conexionFinal) or die (mysql_error());
$cons_entrega=$obj_entrega->consulta();
```

CONSULTAS SQL

CLASES APLICADAS

Este ejemplo solo es el principio de cada módulo ya que hay archivos que pueden llegar a tener 23 consultas de un solo modulo, el cambio que se realizo es muy notorio como podemos observar cada vez que se llama una función se crea un objeto el cual después se complementan las consultas con las funciones mencionadas anteriormente, para obtener los datos requeridos.

4.4.4 Inspección y adaptación. Este bloque de tareas se puede decir que fue de los más complicados y extensos del proyecto pues era una actividad muy mecánica y de concentración ya que se manipulo la parte del modelo lo que representa el módulo en el sistema.

Este bloque de tareas tuvo que pasar por unas pruebas, donde por seguridad se tomó la medida de hacer pruebas de funcionamiento modulo por modulo para ver su funcionamiento, hubo fallos los cuales eran por malos llamados en las variables que se utilizaba al implementar las clases, pero tuvo solución inmediata.

4.5 TAREA 5: PRUEBAS DE FUNCIONAMIENTO DE CADA MODULO SIN IMPLEMENTAR LAS NUEVAS VERSIONES.

4.5.1 Adaptación. Realmente cada vez que se terminaba una tarea, la metodología Scrum nos resalta al hacer una reunión donde se lleva a cabo una inspección y adaptación de la tarea implementada, sin embargo, esta tarea es necesaria realizarla ya que el ingeniero a cargo debe estar completamente seguro que las modificaciones que se realizaron no alteren el comportamiento de los módulos.

4.5.2 Planificación de la iteración. Esta prueba fue necesaria hacerla de manera inmediata pues se implementará el *SOST* modificado en la empresa para ver su comportamiento y las fallas. También se debe implementar lo antes posible ya que el *SOST* está en constante adaptaciones y modificaciones, por lo tanto, si no se

instala de una vez, el ingeniero a cargo trabajara sobre los antiguos archivos y perdemos el trabajo realizado. Ya que toca volver a implementar todos los procedimientos realizados en el módulo que el ingeniero manipule.

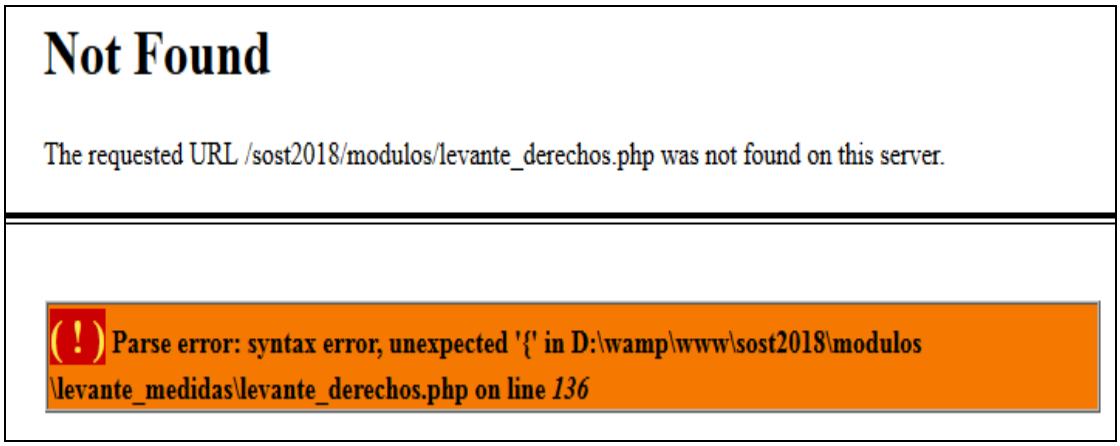
Una vez el *SOST* modificado sea aceptado se actualizarán las versiones y se van hacer las pruebas pertinentes para la actualización.

4.5.3 Ejecución de la iteración. Como es necesario volver a mencionarlo cada cambio realizado no se hizo de manera masiva ya que se podía poner en duda el funcionamiento de cada módulo, por seguridad los cambios se realizaron uno por uno.

por lo cual cada módulo se probó para ver su efectividad y su buen funcionamiento para ello se realizó una lista de requisitos dependiendo la funcionalidad del módulo. El *SOST* contiene varias interfaces diferentes en los módulos, y se implementan según el funcionamiento del mismo, ya que hay unos que repiten la misma interfaz, pero NO su funcionalidad.

Estando en esta etapa se pudieron evidenciar muy pocos errores ya que la idea era avanzar e ir solucionando al final de cada bloque y la mayoría de los errores se provocaron por la reestructuración de los módulos ya que, al agregar una carpeta nueva, implementar clases la nueva, la organización quedaba distinta por lo cual las rutas estaban mal direccionadas y por tal motivo salían los errores.

ILUSTRACIÓN 56. ERRORES COMUNES.



La mayoría de estos errores ocurrían al llamar a un archivo distinto a 3 archivos fundamentales del (M.V.C), como un reporte o una carta requerida, los cuales son archivos dependientes de las funciones y procesos que tienen los módulos, por ello ocurrían los errores tan recurrentes por la ubicación de los archivos y sus malas rutas direccionadas, también al implementar las clases de *MySQL* ya que no se conocía a fondo la codificación de cada módulo y al tener módulos con mas de veinte consultas, ocurrían cruces de variables donde ocurrían los los errores de *Php* por sintaxis.

4.5.4 Inspección y adaptación. Al revisar los módulos con las pruebas establecidas de forma individual (Modulo por Modulo) se iban registrando en una tabla por medio de una lista de chequeado el cual se puede observar en los ANEXOS A.

La actualización de las versiones se implementará una vez software este adaptado en la empresa y con la certeza que su funcionamiento no altere los resultados de los procesos.

Después de tener las pruebas realizadas y estar en un 85% seguros en la funcionalidad de las nuevas modificaciones ya que el 15% restante se va a comprobar tan pronto se implante en la empresa y se compruebe su comportamiento para estar tener un 100% de la funcionalidad de software *SOST*.

Para presentarlo en la empresa se realizó una capacitación debido que la interfaz no va hacer familiar a los funcionarios de la empresa, los temas de la capacitación fueron puntuales era enseñarles a los funcionarios, la nueva interfaz de *SOST* para no tener inconvenientes al momento de aplicarlo, se explicó por qué se cambió la interfaz, se resaltó que los procedimientos de los módulos seguían siendo los mismo a lo que siempre realizaban, también se enseñó lo nuevo que pueden encontrar como el sistema de *Alert*as, los botones, el chat y la interfaz en general.

ILUSTRACIÓN 57. CAPACITACION *SOST* 2018.

4.6 TAREA 6: PRUEBAS DE RENDIMIENTO IMPLEMENTANDO LAS NUEVAS VERSIONES.

4.6.1 Adaptación. Se propuso implementar las nuevas versiones (*Php, MySQL, Apache*) era uno de nuestros objetivos y para ello se realizaron los cambios pertinentes para tal ejecución, ya se tenía estudiado y analizado los cambios a realizar para la actualización, se está llevando a cabo el último bloque de tareas del proyecto.

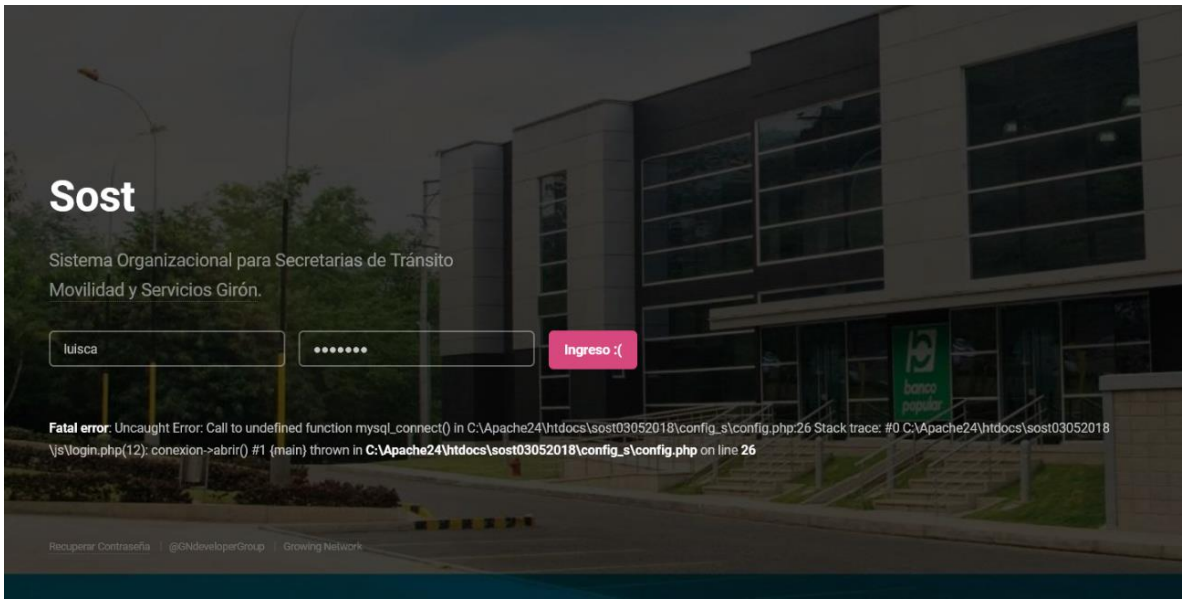
4.6.2 Planificación de la iteración. Esta planificación se puede decir que ha sido la más estudiada y mantenido en práctica en el trascurso del proyecto ya que

todos los cambios, la reestructuración se realizo fue pensando en una actualización tecnológica de las versiones de programación, se instalar las nuevas versiones y adaptarlas ya que el *SOST* maneja una variedad de funciones que son obsoletas para las versiones más avanzadas las cuales ya se han ido arreglando a la medida que se estaba desarrollando el proyecto.

Esta ejecución se está haciendo en un servidor aparte donde se instalaron las últimas versiones de (*Apache, MySQL, Php*) se realizarán las pruebas pertinentes del rendimiento y de seguridad.

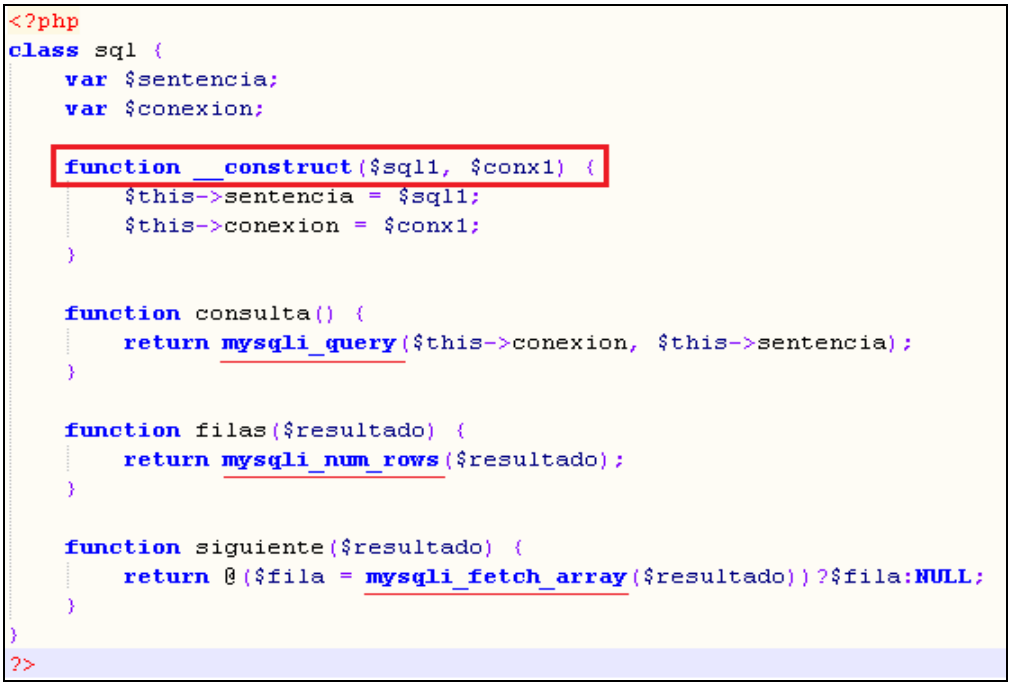
4.6.3 Ejecución de la iteración. Al implementar las nuevas versiones se observaron varios cambios y los mas relevantes fueron los producidos en Php, al implementar el *SOST* actualizado se presentaron varios errores donde se explicarán a continuación:

ILUSTRACIÓN 58. CAMBIO DE NOMBRE DE FUNCIONES.



A primera vista se observa que nos indica el error en el *login,* donde las funciones que contengan *MySQL* como el mysql_connect, el cual se utiliza en el *login* debe ser cambiada por mysqli_connect uno de los principales cambios de las funciones la cual podremos observar en la ILUSTRASION 59.

ILUSTRACIÓN 59. NUEVO ARCHIVO DE CLASES.



```php
<?php
class sql {
    var $sentencia;
    var $conexion;

    function __construct($sql1, $conx1) {
        $this->sentencia = $sql1;
        $this->conexion = $conx1;
    }

    function consulta() {
        return mysqli_query($this->conexion, $this->sentencia);
    }

    function filas($resultado) {
        return mysqli_num_rows($resultado);
    }

    function siguiente($resultado) {
        return @($fila = mysqli_fetch_array($resultado))?$fila:NULL;
    }
}
?>
```

Uno de los bloques realizados consistía en unificar las consulta *SQL* en clases para cuando se actualizara las nuevas versiones solo consistía en cambiar las funciones antiguas que pida la actualización por las nuevas funciones, en este proyecto se tuvo inconvenientes con las clases ya que cuando se creó el archivo que contenía las clases de las funciones de Php para realizar las consultas a la base de datos como lo muestra la ILUSTRASION 52 se observa que el nombre de la clase es igual al constructor que vendría siendo la función principal. En Php versión 7 el nombre de la clase debe ser nombrada como el constructor, como se observa en el recuadro rojo de la ILUSTRASION 59, ese fue uno de los cambios a realizar para poder utilizar la versión

Otro de los cambios que aplico Php fue implementar en los requisitos de conexión a la base de datos el puerto donde se iría a implementar.

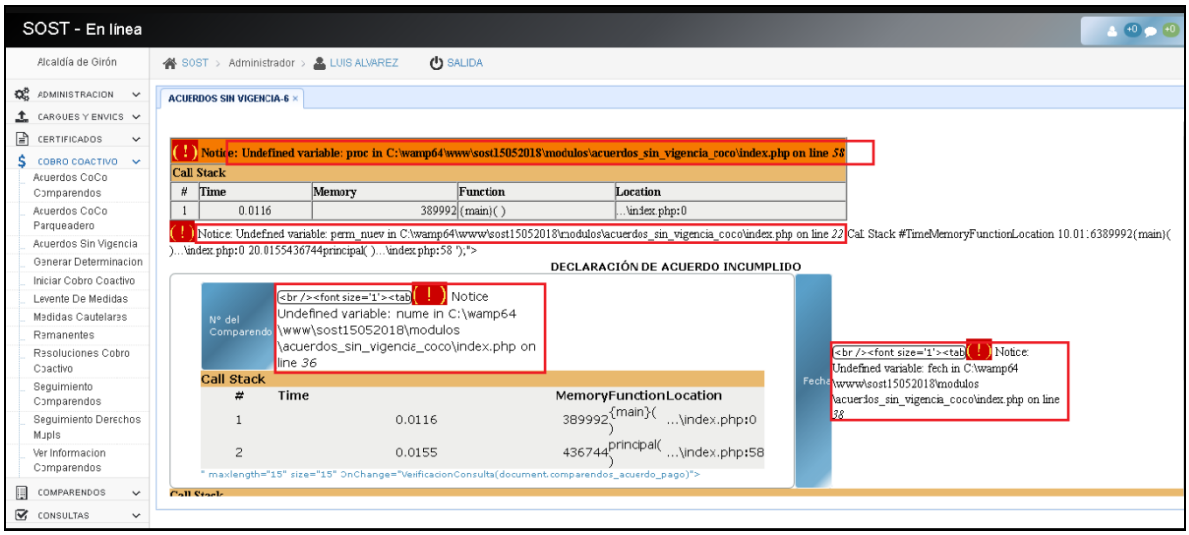
ILUSTRACIÓN 60. IMPLEMENTACION

```php
<?php
date_default_timezone_set('America/Bogota');    setlocale(LC_TIME, 'spanish');
class conexion{
 var $dbhost;
 var $dbunam;
 var $dbpass;
 var $dbname;

 function __construct() {
   $this->dbhost="localhost";
   $this->puerto = "3306";
   $this->dbunam="root";
   $this->dbpass="";
   $this->dbname="sost_m";
 }

    function abrir() {
        $conexion = mysqli_connect($this->dbhost, $this->dbunam, $this->dbpass, $this->dbname,$this->puerto);
        if (!$conexion)  {
            die("Estamos presentando problemas de conexión,  pronto resolveremos el inconveniente   ");
            }
        return $conexion;
    }
}
```

También se comprobó que es susceptible a los espacios en blancos generados por *HTML.*

Como varios de los formularios poseen la misma estructura, mas no su funcionalidad se tienen variables repetidas en algunos módulos, donde unos las utilizan y otros no, estas variables al no estar definidas en la nueva actualización generan errores, ya que en el Php versión 5 no ocurría eso, ya que ignoraba las variables que no estaban definidas siempre y cuando esas variables no se involucraran con la funcionalidad.

ILUSTRACIÓN 61. ERROR POR VARIABLES NO DEFINIDAS.



4.6.3.1 Prueba De Rendimiento Php.

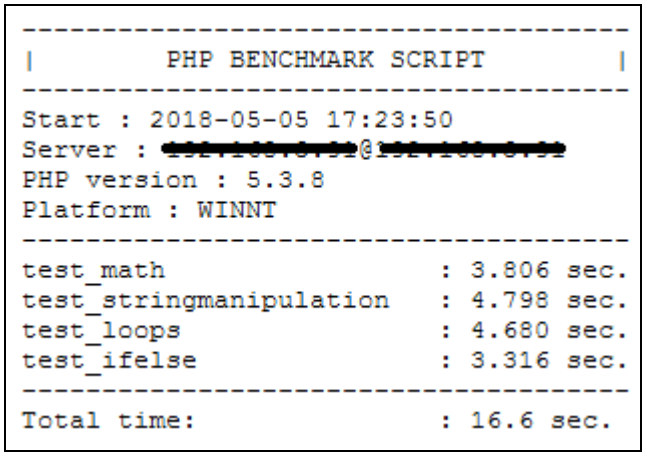
- ***PHP Benchmark Script*:**

Se realizó la instalación correspondiente en un servidor de prueba y se adaptó un script de evaluación de rendimiento de Php el cual es gratuito para calcular las velocidades de referencia (tiempos de ejecución de *Php*) de los servidores el cual se obtuvo de *Php Benchmark Script*[10]. Solo es introducir el script en los archivos del servidor e introducir el nombre en la ruta y el solo realiza el proceso, el cual realiza algunas funciones simples de manipulación de cadenas matemáticas de forma repetitiva, y registra el tiempo de ejecución del código *Php* que se necesita para completar las funciones de *Php*. Para realizar esta prueba y comprobar la diferencia entre las dos versiones a comparar se hizo de manera individual el

[10] ALESSANDRO TORRISI (2010). Php Benchmark Script. [en línea] Disponible en: < https://tools.pingdom.com/ >.

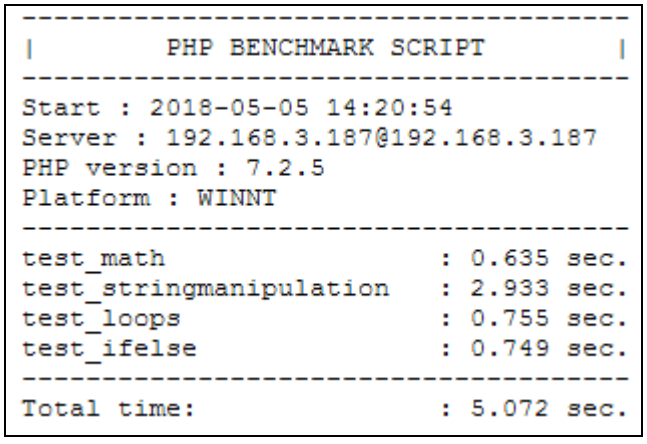
SOST actualmente está manejando una versión de Php 5.3.8 y el cambio es muy grande ya que se instaló una versión de php 7.2.5

FIGURA 7. RESULTADOS DE SCRIPT VERSION PHP 5.3.8



Por políticas de la empresa se eliminó la visibilidad de la Ip de la imagen para mantener la integridad de la información de la empresa. Como se puede observar estos son los resultados de que arroja el script que se ejecutó donde los procesos eran funciones matemáticas, bucles y cadenas de manipulación para poder medir bien el tiempo de ejecución.

FIGURA 8. RESULTADOS DE SCRIPT VERSION PHP 7.2.5



Como se puede observar es un servidor de prueba ya que se puede mostrar su IP y también se puede evidenciar la gran diferencia de tiempos de ejecución son casi 10 segundos de diferencia en ejecución del script de prueba. Se puede aclarar que era fundamental el cambio de versión ya que nos va asegurar más rapidez por parte de *Php*.

4.6.3.1 Prueba de Velocidad Web. Esta prueba se realiza a páginas web para ver la velocidad al cargar y las falencias obtenidas, también hace recomendaciones para mejorar su rendimiento ya que esas pruebas se hacen en páginas especificas las cuales direccionan a un servidor donde hace el diagnostico.

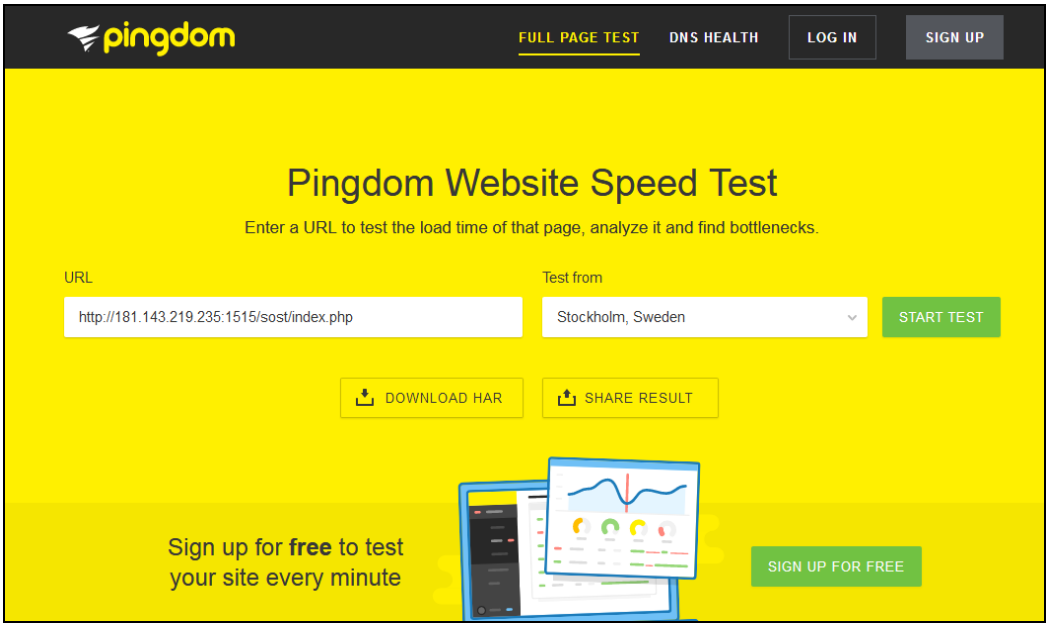
Para poder hacer la prueba era necesario pasar la *Ip* de direccionamiento del servidor de la empresa y el de prueba a públicas para que se puedan observar en la web externa, estas pruebas se realizaron en dos páginas distintas para tener varios puntos de vista y hacer bien las comparaciones.

- **PINGDOM:**

Esta página ofrece variedad recursos como: monitoreo de tiempo de actividad, control de velocidad de página, monitorear servidores y aplicaciones web, y se encarga de ofrecer el servicio de supervisión en el sitio web a partir de datos reales[11]. Se introducido los links de los dos *SOST* el actualmente y con el de las versiones actualizadas, con su respectiva *Ip* publica y realizo el diagnóstico sobre el rendimiento general de la web y muestra una prueba de velocidad y si se detectan problemas**.**

[11]PINGDOM (2018). Pingdom Website Speed Test. [en línea] Disponible en: < https://tools.pingdom.com/ >.

ILUSTRACIÓN 62. PRUEBA #1 *SOST* ORIGINAL.



Las *Ip* son públicas solo para poder realizar las pruebas en los servidores que ofrecen el servicio, después de realizar las pruebas volverán a su *Ip* privada para el funcionamiento interno de la empresa.

ILUSTRACIÓN 63. PRUEBA #1 *SOST* ACTUALIZADA.

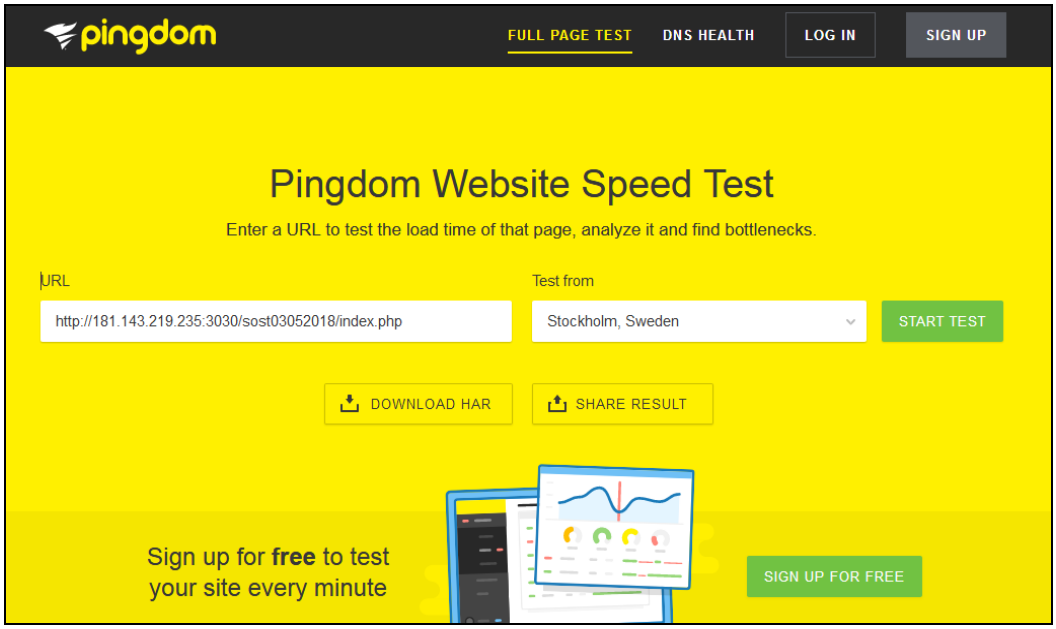
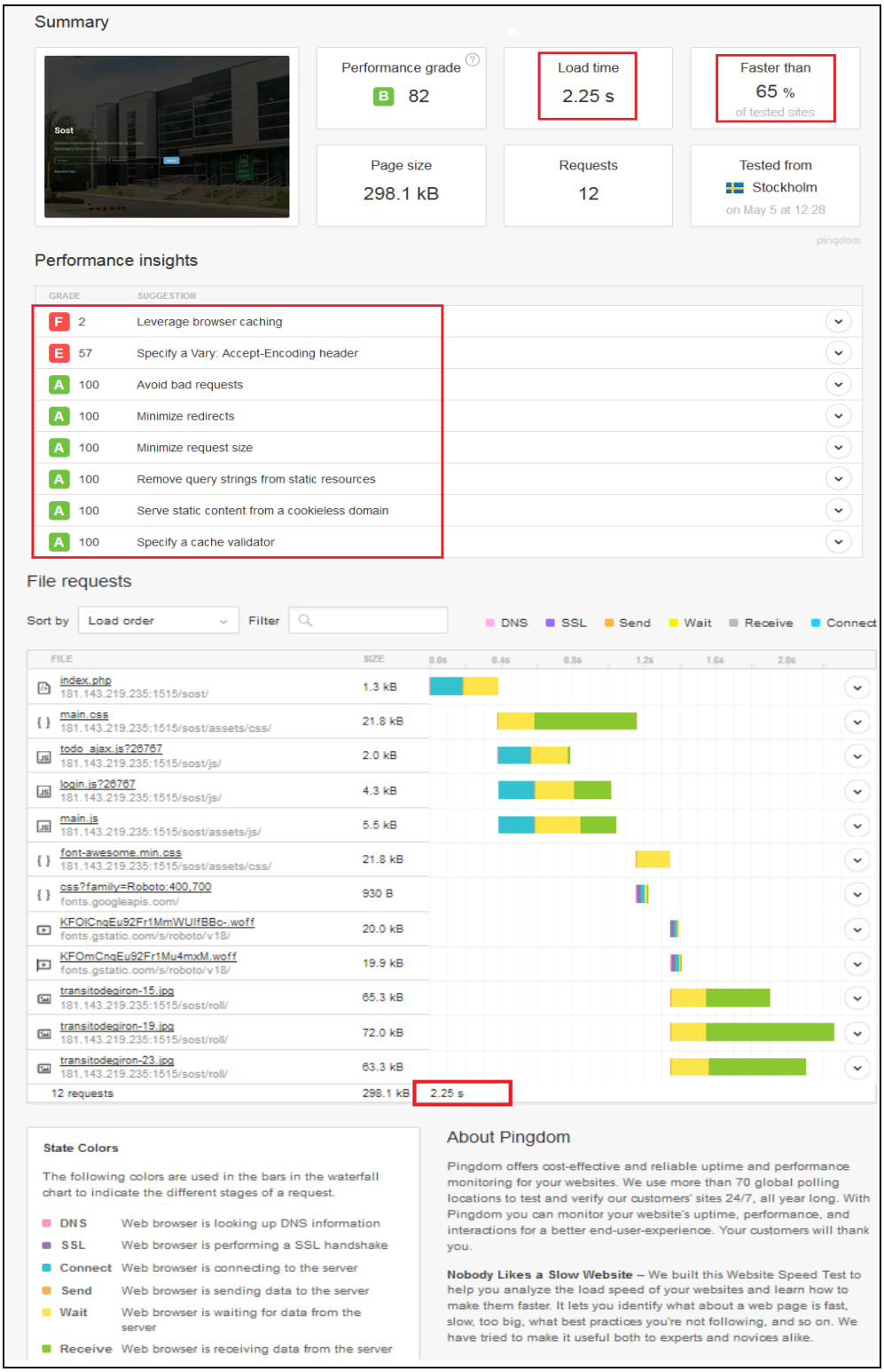


ILUSTRACIÓN 64. RESULTADOS #1 *SOST* ORIGINAL.



Como se observa en la imagen es el resultado obtenido del análisis de la página muestra unos datos relevantes los cuales se encuentran en un cuadro rojo nos indica que:

- El tiempo de cargar los archivos fue de 2.25 segundos.
- Dice que es el 65% más rápida de todas las páginas web que han utilizado el respectivo análisis
- Indica que de las 8 pruebas solo se quedó en 2, las cuales nos recomienda utilizar más la memoria de cache para hacer así la página más rápida.

También nos muestra los datos como el grado de rendimiento es un 82, lo cual es muy bueno, ya que así es el rendimiento que se está manejando actualmente en la empresa también indica que esta prueba fue realizada en un servidor en Estocolmo.

Las falencias obtenidas, la página las muestra explícitamente para tener en cuenta el arreglo y hacer la página más rápida.

ILUSTRACIÓN 65. FALENCIAS OBTENIDAS *SOST* ORIGINAL ANALISIS #1.

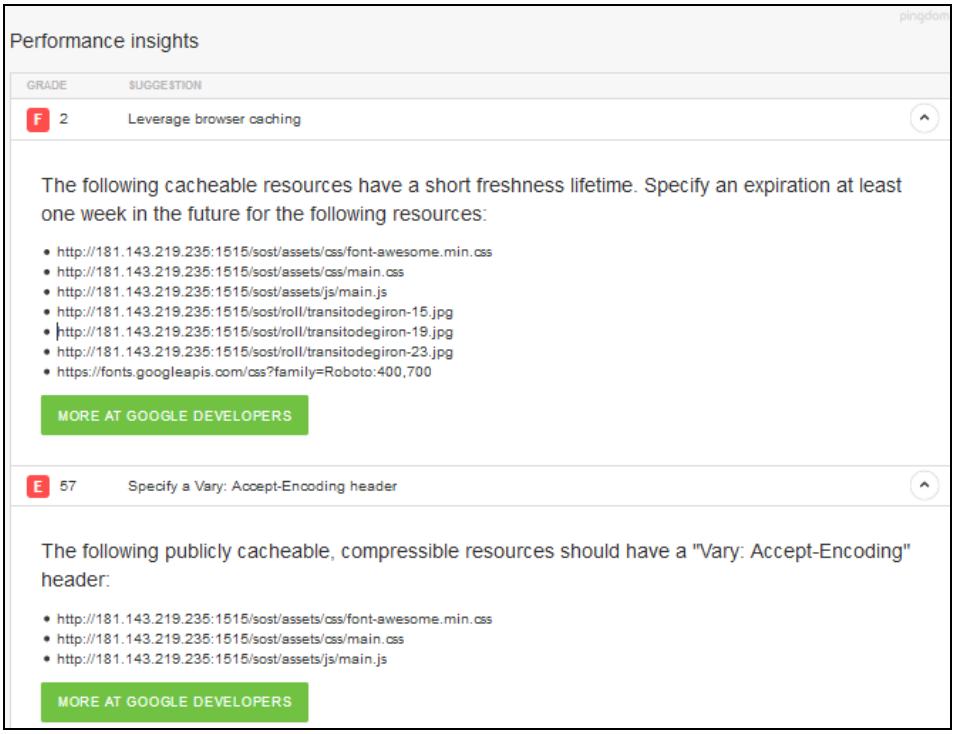
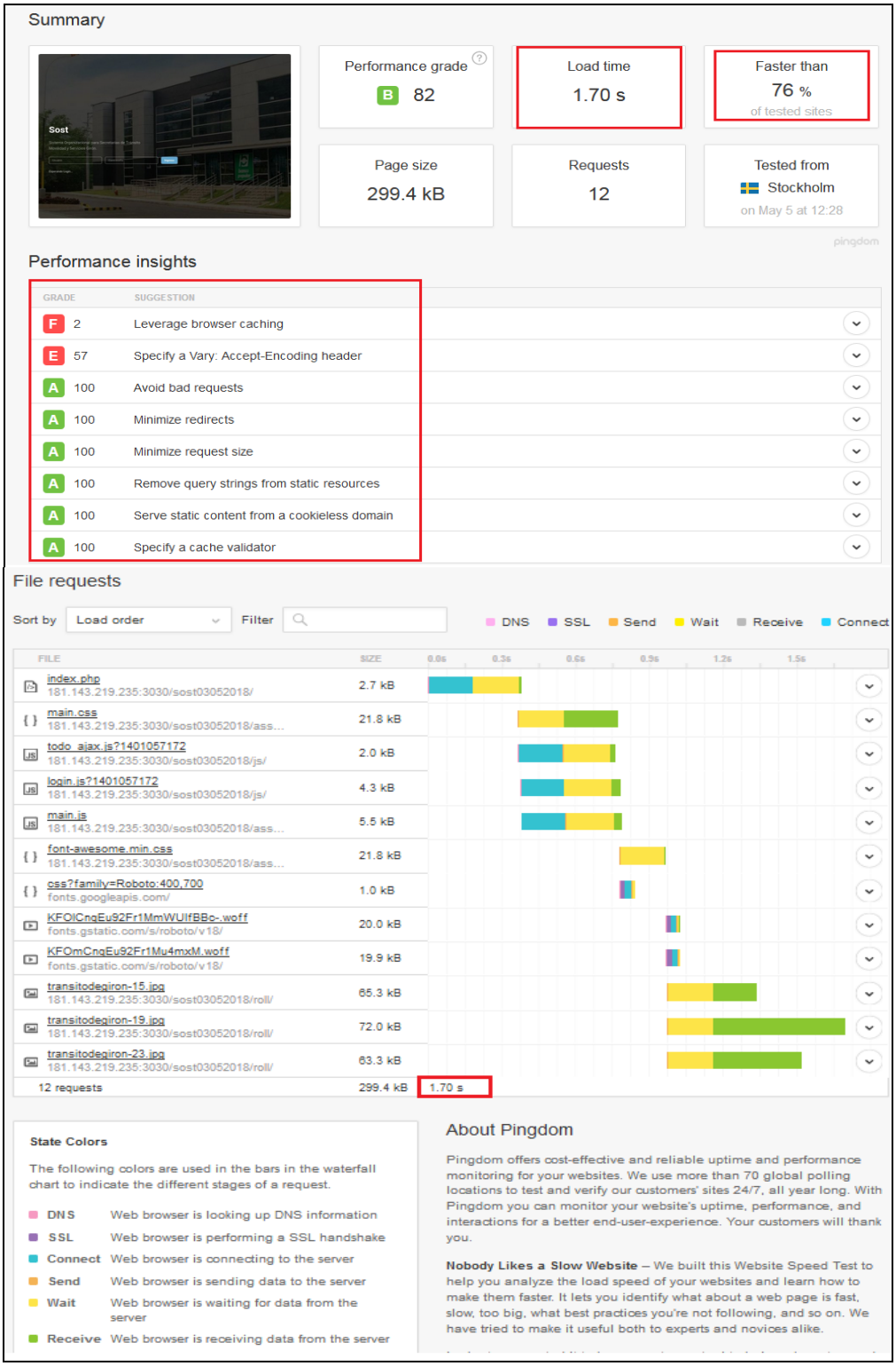


ILUSTRACIÓN 66. RESULTADOS #1 *SOST* ACTUALIZADO.



La imagen anterior enseña los resultados de la prueba con las herramientas actualizadas donde se actualizo *(Apache*, *MySQL*, Php). Se puede hacer una comparación con la ilustración 60 donde es el mismo software, pero con las versiones desactualizadas. Este análisis nos muestra una gran diferencia con el software original, estas se pueden destacar en el recuadro rojo que indica:

- El tiempo de cargar los archivos fue de 1.70 segundos, se puede decir que es 0.5 segundo más rápido que el original.
- Dice que es el 76% más rápida de todas las páginas web que han utilizado el respectivo análisis, comparándola es un 11% más rápida.
- Indica que de las 8 pruebas solo se quedó en 2, las cuales nos recomienda utilizar más la memoria de cache para hacer así la página más rápida, este resultado dio igual que el análisis anterior ya que la estructura es igual, solo se actualizaron las versiones.

También como el análisis anterior, muestra los datos como el grado de rendimiento es un 82, lo cual sigue siendo muy bueno, ya que así es el rendimiento que se está manejando actualmente en la empresa, para hacer equitativa el análisis también se realizó en un servidor de Estocolmo.

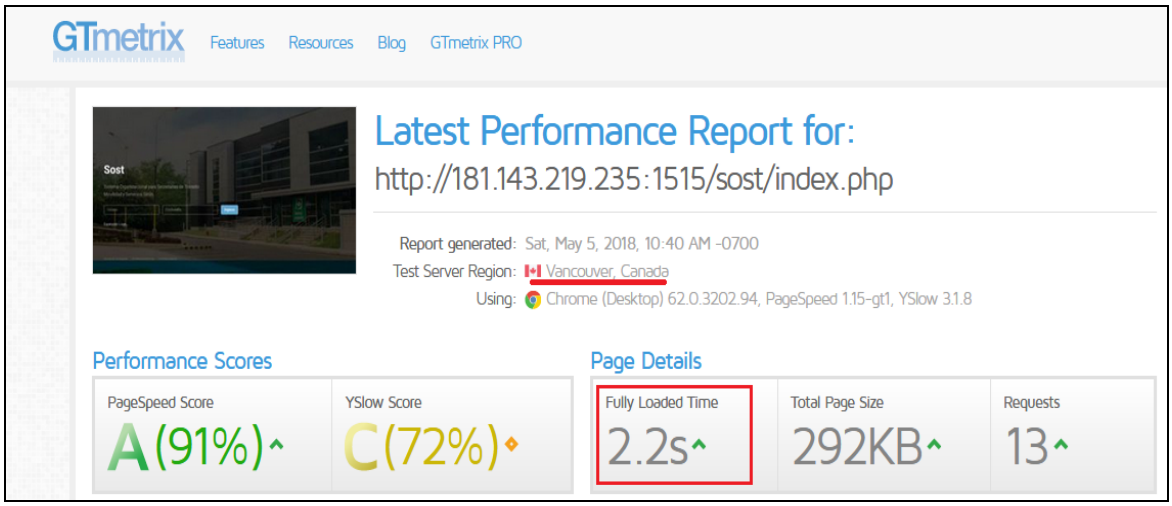
Las falencias obtenidas, en este análisis fueron las mismas que el anterior ya que se debe mejorar al utilizar más la memoria de cache para hacer que no esté cargando, datos que nunca van a tener una modificación, como lo indica la ilustración 61.

También se realizaron las pruebas en otra página que ofrece el servicio de análisis de nuestro portal web en este caso el *SOST* también tendría que utilizar la ip publica para poder realizar el análisis.

- **GTMETRIX:**

Esta página analiza la velocidad de tu sitio y aconseja para volverlo más rápido. GTmetrix le da una idea de qué tan bien se carga su sitio y brinda recomendaciones prácticas sobre cómo optimizarlo[12].

ILUSTRACIÓN 67. ANALISIS #2 *SOST* ORIGINAL.



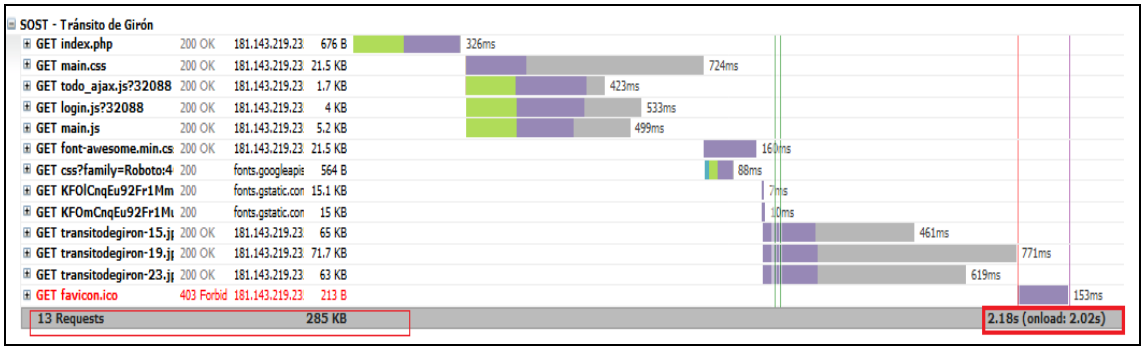
Como se puede observar el proceso del análisis es igual que el anterior análisis, ya que solo basta con introducir el link y la pagina direcciona a un servidor. Podemos observar que los valores son muy parecidos al análisis que genero el anterior estudio teniendo en cuenta que esta imagen son los datos del *SOST* que actualmente está trabajando la empresa

Nos indica que el tiempo completamente cargando es de 2.2 segundos también muestra que la pagina es un 91% más rápido comparándola con otros portales

[12]GTMETRIX (2018). Analyze [en línea] Disponible en: < https://gtmetrix.com/analyze.html >.

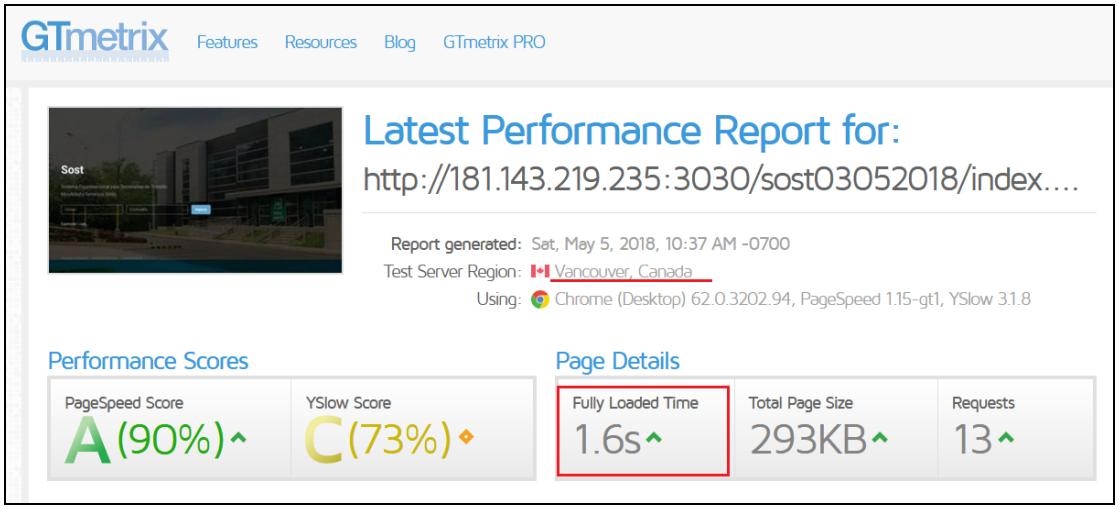
web que han hecho el análisis en esta misma página. Este análisis se realizó en un servidor ubicado en Vancouver.

ILUSTRACIÓN 68. LÍNEA DE TIEMPO PRUEBA #2 *SOST* ORIGINAL



Podemos observar la línea del tiempo de ejecución de cada archivo del *SOST*, el tiempo que se demora para poder completar la carga completa de la página.

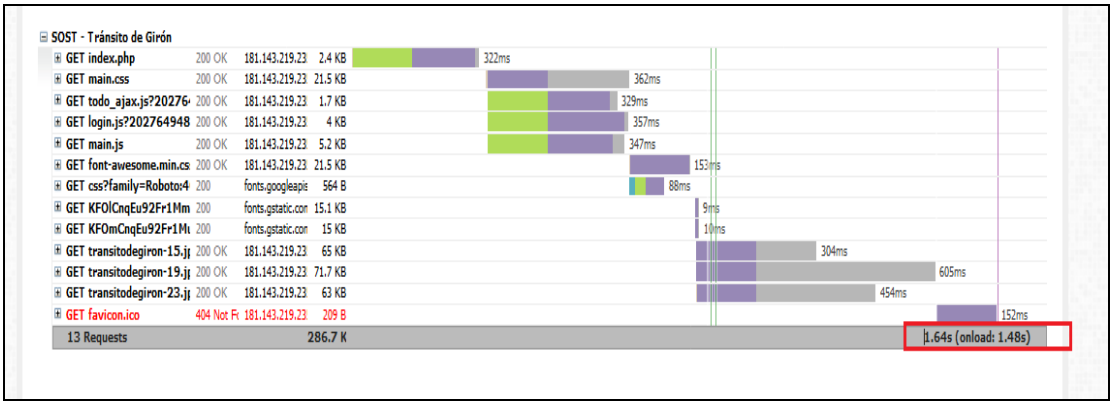
ILUSTRACIÓN 69. ANALISIS #2 *SOST* ACTUALIZADO.



Esta prueba se realizó al *SOST* actualizado, como lo indica el resultado encerrado en un cuadro de color rojo muestra que la velocidad es de 1.6 segundos donde es más velos que el *SOST* original por 0.6 segundos, este análisis nos confirma que las actualizaciones de las versiones eran necesarias ya que brindan más velocidad al ejecutar funciones en el *SOST* actualizado. En este caso tuvimos un desfase ya que dice que es el 90% más rápida de todas las pruebas realizadas en

el sitio web, cuando el *SOST* original tiene un 91% de todos modos en un muy buen porcentaje de comparación.

ILUSTRACIÓN 70. LÍNEA DE TIEMPO PRUEBA #2 *SOST* ACTUALIZADO.



Esta página nos proporciona la línea de tiempo de carga de los archivos que contenga la página web como se ilustró en el análisis del *SOST* original se pueden comparar los tiempos de ejecución.

Por conclusión en estas tres pruebas de rendimiento realizadas, se verifican y se comprueban que la actualización era necesaria ya que brinda más velocidad de ejecución, sin contar las nueva funciones y adaptaciones que traen las nuevas versiones por defecto como mayor seguridad y garantizar la escalabilidad del software.

5. CONCLUSIONES

Culminado el proyecto se puede concluir que:
Es una oportunidad de mejoramiento empresarial de gran beneficio, por lo tanto, fue necesario que el *SOST* siendo una herramienta fundamental tuviera actualizaciones tecnológicas en cuanto a las versiones de programación que implementa, ya que se consideró la parte funcional como la interfaz del software.

Este proyecto es un ejemplo que, al mantener un software en un buen estado actualizable, no deja que la funciones y clases se vuelvan obsoletas. Se garantiza una escalabilidad permanente para poder avanzar en las actualizaciones, teniendo el software con las últimas versiones de las herramientas que utilizan para su funcionamiento, nos certifica que va a mejorar su velocidad en cuestión de cargar sus formularios y consultas, también la seguridad aumenta manteniendo la confidencialidad de la información.

El trabajo de grado en la modalidad de Practica Empresarial es una buena oportunidad de actualizar el aprendizaje, implementando de manera practica el conocimiento obtenido. La práctica empresarial permite interactuar con expertos, analizando errores y aprendiendo de ellos. Adicionalmente se aprende a obtener resultados en los tiempos establecidos por el cronograma, verificando el cumplimiento explícito de los objetivos.

6. RECOMENDACIONES

Recomendaciones según el proyecto, donde también se puede generalizar aclarando que todo proyecto software, o todo proyecto que esté relacionado con tecnología debe estar en continua actualización y mejora, se sugieren a continuación las recomendaciones para *SOST*:

- Capacitar a funcionarios que ingresan recientemente a la empresa, a desarrollar algún trabajo donde se involucre la manipulación del *SOST*.
- Aprovechar la escalabilidad, para continuar con los avances tecnológicos en el software, siempre y cuando lo apruebe el encargado del área de sistemas.
- Proponer cambiar la capacidad del *hardware* en cuanto a la RAM del servidor, para también aumentar el rendimiento de los procesos del sistema.
- Considerar en el plan de estudios de Ingeniería de Sistemas, tomar una orientación más practica en las clases donde se maneja bastante teoría, para así envolver al estudiante al mudo laboral.

BIBLIOGRAFÍA

ALESSANDRO TORRISI (2010). Php Benchmark Script. [en línea] Disponible en: < https://tools.pingdom.com/ >.

APACHE HTTP SERVER. (1997) ¿Qué es apache? [en línea] Disponible en: < https://wiki.apache.org/httpd/FAQ#What_is_Apache.3F >.

EORTS (2017). Programación: JavaScript. [en línea] Disponible en: < http://eorts.com/javascript.html >.

GTMETRIX (2018). Analyze [en línea] Disponible en: < https://gtmetrix.com/analyze.html >.

HPE (2018). SERVIDORES: Servidores ProLiantServidor: HPE ProLiant ML110 Gen10 [en línea] Disponible en:<https://www.hpe.com/us/en/product-catalog/servers/proliant-servers/pip.hpe-proliant-ml110-gen10-server.1010192782.html>.

LIBROSWEB (2006). Introducción a *AJAX*: Capítulo 1. Introducción a *AJAX*. [en línea] Disponible en: < http://librosweb.es/libro/*Ajax*/capitulo_1.html>.

MOVILIDAD Y SERVICIOS GIRON S.A.S. (2012). Tránsito de girón: nosotros [en línea] Disponible en: <http://www.transitodegiron.com.co/> [citado 14 de diciembre de 2011].

PINGDOM (2018). Pingdom Website Speed Test. [en línea] Disponible en: < https://tools.pingdom.com/ >.

PROYECTOS AGILES. Qué es SCRUM [en línea] Disponible en:< https://proyectosagiles.org/que-es-scrum/>.

UNIVERSIDAD DE ALICANTE. Servicio de Informática ASP.NET MVC 3 Framework: Modelo vista controlador (MVC) [en línea] Disponible en: < https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html >.

WIKIPEDIA (2018). *MySQL* [en línea] Disponible en: <https://es.wikipedia.org/wiki/*MySQL*>.

ANEXOS

ANEXO A. Documentación de pruebas realizadas a cada módulo.

MODULOS	FUNCIONES										
	AGREGAR	ELIMINAR	BUSCAR	EDITAR	GUARDAR	CARGAR	VISTA PREVIA	EMIGRAR	REPORTES	IMPRIMIR	VALIDAR DEUDA
Anular Recibos - Administracion	NA	NA	NA	NA	OK	NA	OK	NA	OK	NA	NA
Perfiles - Administracion	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA	NA
Archivos Simit Y Est. - Cargues Y Envios	NA	NA	NA	NA	NA	OK	OK	OK	OK	OK	OK
Cargue Derechos PSE - Cargues Y Envios	OK	NA	NA	NA	NA	OK	NA	NA	NA	NA	NA
Pagos Simit - Cargues Y Envios	OK	NA	NA	NA	NA	OK	NA	NA	OK	NA	NA
Certificado Tradicion - Certificados	NA	NA	NA	NA	OK	OK	NA	NA	NA	OK	OK
Acuerdos CoCo Comparendos - Cobro Coactivo	NA	NA	NA	NA	OK	OK	OK	NA	NA	OK	NA
Iniciar Cobro Coactivo - Cobro Coactivo	OK	NA	NA	NA	NA	NA	NA	NA	OK	NA	NA
Remanentes - Cobro Coactivo	OK	NA	OK	OK	OK	OK	OK	NA	NA	NA	NA
Seguimiento Comparendos - Cobro Coactivo	NA	NA	NA	NA	NA	OK	OK	OK	OK	OK	NA
Ver Informacion Comparendos - Cobro Coactivo	NA	NA	NA	NA	NA	OK	NA	NA	NA	NA	NA
Comparendos - Comparendos	NA	NA	OK	NA	NA	OK	OK	NA	NA	NA	NA
Editar Audiencias - Comparendos	OK	NA	NA	OK	OK	OK	NA	NA	NA	NA	NA
Libretas - Comparendos	OK	NA	OK	NA	OK	NA	OK	NA	NA	NA	NA
Consulta Caracteristica - Consultas	NA	NA	NA	NA	OK	OK	OK	OK	OK	OK	NA
Consulta SYC - Consultas	NA	NA	OK	NA	NA	OK	OK	NA	NA	NA	NA
Carpetas Pedidas - G. Documental	NA	NA	NA	NA	NA	NA	OK	NA	NA	NA	NA
Entrega De Placas - G. Documental	OK	NA	OK	NA	OK	NA	OK	NA	NA	NA	NA
Guardar Ubicacion - G. Documental	OK	NA	OK	NA	OK	NA	OK	NA	NA	NA	NA
Pedir Carpetas - G. Documental	NA	NA	OK	NA	NA	OK	OK	OK	OK	OK	NA
Revision De Control Expedientes - G. Documental	NA	NA	OK	NA	NA	OK	OK	NA	OK	NA	NA
Comparativo Personas-Placas - Juridica	NA	NA	NA	NA	OK	OK	OK	NA	NA	OK	NA
Reporte Estado De Cuenta - Juridica	NA	NA	NA	NA	OK	OK	OK	OK	OK	OK	NA
BANCO Reporte - Liquidaciones	NA	NA	OK	NA	NA	OK	OK	OK	OK	OK	NA
Liquidar Derechos Mpls - Liquidaciones	NA	NA	OK	NA	NA	OK	NA	NA	OK	OK	NA
Procesos Especiales - Liquidaciones	NA	OK	OK	NA	OK	OK	OK	NA	NA	NA	NA
Bancos - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Clases - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Conceptos - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Division Politica - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Escuela Gestor Crc - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Instituciones - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Modalidades - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Organismos De Transito - Mantenimiento	OK	NA	OK	OK	OK	OK	OK	NA	NA	NA	NA
Salarios - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Tipo Identificaciones - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Tipo Pasajeros - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Actos Administrativos - Matriculas	OK	NA	OK	OK	OK	OK	OK	NA	NA	NA	NA

Generar Acto Administrativo - Matriculas	NA	NA	OK	NA	NA	OK	OK	NA	OK	OK	NA
Licencias De Transito - Matriculas	NA	NA	OK	NA	NA	NA	OK	OK	OK	OK	NA
Registro Inicial - Matriculas	OK	NA	NA	OK	OK	OK	NA	NA	NA	NA	NA
Salida De Vehiculos - Parqueadero	NA	NA	NA	NA	OK	NA	NA	NA	NA	NA	NA
Reporte Estado Concesionarias - Reportes	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	NA
Reporte Inspeccion - Reportes	NA	NA	OK	NA	NA	OK	OK	OK	OK	OK	NA
Reportes Coactivo - Reportes	NA	NA	OK	NA	NA	OK	OK	OK	OK	OK	NA
Rangos Licencias Conduccion - Tr. Conductores	OK	NA	OK	NA	OK	OK	NA	NA	NA	NA	NA
Cambio Color - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Cambio Placas - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Duplicado Licencia Transito - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Levantamiento Alerta - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Regabar Serie - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Sucesiones - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Traspaso - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Dias Habiles - Administracion	NA	NA	NA	OK	OK	NA	OK	NA	NA	NA	NA
Permisos - Administracion	NA	NA	NA	OK	OK	NA	OK	NA	NA	NA	NA
Tarifas - Administracion	OK	NA	NA	OK	OK	NA	OK	NA	NA	NA	NA
Cargue De Comparendos - Cargues Y Envios	OK	NA	NA	NA	OK	OK	NA	NA	NA	NA	NA
Cargue Derechos REF - Cargues Y Envios	OK	NA	NA	NA	OK	OK	NA	NA	NA	NA	NA
Certificado Paz Y Salvo - Certificados	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Certificados Varios - Certificados	NA	NA	OK	NA	NA	OK	OK	NA	NA	NA	NA
Acuerdos CoCo Parqueadero - Cobro Coactivo	NA	NA	OK	NA	NA	OK	OK	NA	OK	NA	NA
Levente De Medidas - Cobro Coactivo	NA	NA	OK	NA	OK	NA	OK	OK	OK	OK	NA
Reporte Data Credito - Cobro Coactivo	NA	NA	OK	NA	OK	OK	OK	OK	OK	OK	NA
Seguimiento Derechos Mupls - Cobro Coactivo	NA	NA	OK	NA	OK	NA	OK	OK	OK	OK	NA
Agentes - Comparendos	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Comparendos Ajuste - Comparendos	OK	NA	NA	NA	OK	OK	NA	NA	NA	NA	NA
Estado Comparendo - Comparendos	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Sancion Y Cobro Coativo - Comparendos	NA	NA	OK	NA	OK	OK	OK	OK	NA	OK	NA
Consulta Comparendos - Consultas	NA	NA	OK	NA	NA	OK	NA	NA	NA	NA	NA
Consulta Tarifas - Consultas	NA	NA	OK	NA	OK	NA	OK	OK	OK	OK	NA
Control De Expedientes - G. Documental	OK	NA	OK	OK	OK	OK	NA	NA	NA	NA	NA
Enviar A Archivo - G. Documental	NA	NA	OK	NA	NA	NA	OK	NA	OK	OK	NA
Historial De Carpeta - G. Documental	NA	NA	OK	NA	NA	NA	OK	NA	OK	OK	NA
Recepcion De Placas - G. Documental	OK	NA	OK	NA	OK	OK	NA	NA	NA	NA	NA
Revision De Digitalizaciones - G. Documental	NA	NA	OK	NA	NA	OK	OK	NA	OK	OK	NA
Pendientes - Juridica	OK	NA	OK	OK	OK	OK	NA	NA	NA	NA	NA
BANCO Pagos Runt - Liquidaciones	OK	OK	NA	OK	OK	NA	NA	NA	OK	OK	NA
Incluir Derechos - Liquidaciones	NA	NA	OK	NA	OK	OK	NA	NA	NA	NA	NA
Liquidar Parqueadero - Liquidaciones	NA	NA	OK	NA	OK	OK	NA	NA	OK	OK	NA
Actualizar Personas - Mantenimiento	OK	NA	OK	OK	OK	OK	OK	NA	NA	NA	NA
Barrios - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA

Colores - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Concesionarios - Mantenimiento	OK	NA	OK	OK	OK	OK	OK	NA	NA	NA	NA
Empresas - Mantenimiento	OK	NA	OK	OK	OK	OK	OK	NA	NA	NA	NA
Estado - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Lineas - Mantenimiento	OK	NA	OK	OK	OK	OK	OK	NA	NA	NA	NA
Niveles - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Puertos - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Servicios - Mantenimiento	OK	NA	OK	OK	OK	OK	OK	NA	NA	NA	NA
Tipo Infractor - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Tipo Recaudo - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Actualizar Matricula - Matriculas	OK	NA	NA	OK	OK	OK	NA	NA	NA	NA	NA
Historial Radicado - Matriculas	NA	NA	OK	NA	OK	OK	OK	NA	NA	NA	NA
Radicado De Cuenta - Matriculas	OK	NA	NA	OK	OK	OK	NA	NA	NA	NA	NA
Seguimiento De Placas - Matriculas	NA	NA	OK	NA	NA	OK	OK	NA	NA	NA	NA
Relacion Comparendos - Reportes*	NA	NA	OK	NA	NA	OK	OK	OK	OK	OK	NA
Reporte General - Reportes	NA	NA	OK	NA	NA	OK	OK	OK	OK	OK	NA
Reporte Inventarios - Reportes	NA	NA	OK	NA	NA	OK	OK	OK	OK	OK	NA
Certificado De Licencias - Tr. Conductores	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Cambio Capacidad - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Cambio De Empresa - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Cambio Servicio - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Duplicado Placa - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Regabar Chasis - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Rematricula - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Tarjetas De Operacion - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Formularios - Administracion	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Pqrs - Administracion	NA	NA	OK	OK	OK	OK	OK	NA	NA	NA	NA
Usuarios - Administracion	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Archivos Migracion - Cargues Y Envios	NA	NA	NA	NA	NA	OK	NA	NA	OK	NA	NA
Cargue De Pendientes - Cargues Y Envios+	NA	NA	NA	NA	OK	OK	NA	NA	OK	NA	NA
Cargue Derechos SYC - Cargues Y Envios	NA	NA	NA	NA	OK	OK	NA	NA	OK	NA	NA
Certificado Propiedad - Certificados	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Devoluciones - Certificados	NA	NA	OK	NA	OK	OK	NA	NA	OK	NA	NA
Generar Determinacion - Cobro Coactivo	NA	NA	OK	NA	NA	OK	OK	OK	OK	OK	NA
Medidas Cautelares - Cobro Coactivo	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Resoluciones Cobro Coactivo - Cobro Coactivo	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Sin Vigencia Acuerdo - Cobro Coactivo	NA	NA	OK	NA	NA	OK	OK	NA	NA	NA	NA
Asignar Libretas - Comparendos	OK	NA	OK	NA	OK	OK	OK	NA	NA	NA	NA
Comparendos Procesos - Comparendos	OK	NA	OK	OK	OK	OK	NA	NA	NA	NA	NA
Inmovilizados - Comparendos	OK	NA	OK	NA	OK	OK	OK	OK	NA	NA	NA
Busqueda VIP - Consultas	NA	NA	OK	NA	NA	NA	NA	NA	OK	NA	NA
Consulta Simit - Consultas	NA	NA	OK	NA	NA	OK	OK	NA	OK	NA	NA
Carpetas Para Archivar - G. Documental	NA	NA	OK	NA	OK	OK	NA	NA	NA	NA	NA
Crear Etiquetas - G. Documental	OK	NA	NA	NA	OK	OK	NA	NA	NA	OK	NA

Envio SMS - G. Documental	OK	NA	OK	NA	OK	OK	OK	NA	NA	NA	NA
Oficios Y Otros - G. Documental	OK	NA	OK	OK	OK	OK	OK	NA	NA	NA	NA
Reportes GD - G. Documental	NA	NA	OK	NA	NA	OK	OK	OK	OK	OK	NA
Autorizacion Pendientes - Juridica	NA	NA	OK	NA	OK	OK	OK	NA	NA	NA	NA
Reporte De Pendientes - Juridica	NA	NA	OK	NA	NA	OK	OK	OK	OK	OK	NA
BANCO Recaudo - Liquidaciones	OK	OK	NA	OK	OK	NA	NA	NA	OK	OK	NA
Liquidaciones Generales - Liquidaciones	NA	NA	OK	NA	OK	OK	OK	NA	NA	OK	NA
Pagos Externos - Liquidaciones	NA	NA	OK	NA	OK	OK	NA	NA	OK	NA	NA
Asignar Placas - Mantenimiento	OK	NA	OK	NA	OK	OK	OK	NA	NA	NA	NA
Carrocerias - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Combustibles - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Devolucion De Radicado - Mantenimiento	NA	NA	OK	OK	OK	OK	OK	NA	NA	NA	NA
Entidades - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Infracciones - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Marcas - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Orden De Pedidos - Mantenimiento	NA	NA	OK	NA	NA	OK	OK	OK	OK	OK	NA
Rangos De Placas - Mantenimiento	OK	NA	OK	NA	OK	OK	OK	NA	NA	NA	NA
Tipo Categoria - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Tipo Organismos - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Vendedores - Mantenimiento	OK	OK	OK	OK	OK	OK	OK	NA	NA	NA	NA
Control Tramite - Matriculas	NA	NA	OK	NA	NA	OK	OK	NA	NA	NA	NA
Hoja De Ruta - Matriculas	NA	NA	OK	NA	NA	OK	OK	NA	NA	NA	NA
Rangos Licencia Transito - Matriculas	OK	NA	OK	NA	OK	OK	OK	NA	NA	NA	NA
Ingresar Inmovilizado - Parqueadero	OK	NA	OK	OK	OK	OK	OK	NA	NA	NA	NA
Relacion De Concesionarios - Reportes	NA	NA	OK	NA	NA	OK	OK	OK	OK	OK	NA
Reporte Ingresos - Reportes	NA	NA	OK	NA	NA	OK	OK	OK	OK	OK	NA
Reporte Tramites Y Matriculas - Reportes	NA	NA	OK	NA	NA	OK	OK	OK	OK	OK	NA
Licencias De Conduccion - Tr. Conductores	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Cambio Carroceria - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Cambio Motor - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Cancelacion Matricula - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Inscripcion Alerta - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Regabar Motor - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Repotenciacion - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK
Traslado De Cuenta - Tr. Vehiculos	NA	NA	NA	NA	OK	OK	NA	NA	NA	NA	OK

NA = No aplica

OK = Correcto

ANEXO B. Cronograma implementado en el proyecto.