UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de:
INGENIERO DE SISTEMAS

TEMA:

ANÁLISIS, DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA WEB PARA LA GESTIÓN DE PRODUCTOS Y SERVICIOS ENFOCADO A PYMES, ENMARCADO EN EL ENTORNO DE UNA VETERINARIA

AUTOR:

DIEGO MARTÍN GARCÍA MOYA

TUTOR:

ROBINSON DIMITRI LLERENA PAZ

Quito, julio del 2019

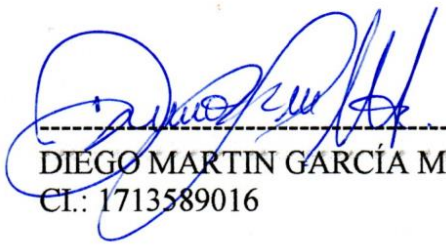
Quito, julio 2019

CESIÓN DE DERECHOS DE AUTOR

Yo, Diego Martín García Moya, con documento de identificación N.- 1713589016, manifiesto mi voluntad y cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del trabajo de titulación con tema: "ANÁLISIS, DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA WEB PARA LA GESTIÓN DE PRODUCTOS Y SERVICIOS ENFOCADO A PYMES, ENMARCADO EN EL ENTORNO DE UNA VETERINARIA", mismo que ha sido desarrollado para optar por el título de INGENIERO DE SISTEMAS en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autor me reservo los derechos morales de la obra antes citada.

En concordancia, suscribo este documento en el momento que hice la entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.



DIEGO MARTIN GARCÍA MOYA
CI.: 1713589016

Quito, julio 2019

DECLARATORIA DE COAUTORÍA DEL TUTOR

Yo, declaro que bajo mi dirección y asesoría fue desarrollado el proyecto técnico, con tema: "ANÁLISIS, DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA WEB PARA LA GESTIÓN DE PRODUCTOS Y SERVICIOS ENFOCADO A PYMES, ENMARCADO EN EL ENTORNO DE UNA VETERINARIA" realizado por DIEGO MARTÍN GARCÍA MOYA, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerados como trabajo final de titulación.

ROBINSON DIMITRI LLERENA PAZ
CI.: 1707106942

DEDICATORIA

Este trabajo de Titulación lo quiero dedicar a mis padres, Doña Yolanda y el Don Rogelio, los mismos que fueron el puntal más fuerte en el desarrollo de mi carrera, entregándome todo su amor y sabiduría; también se la dedico a mi esposa Tatiana, quien siempre supo comprenderme y apoyarme en todo momento, a mis hijos Diego Andrés y Carolina Abigaíl, por su infinito amor, y finalmente, a mi hermana Tulia, por estar siempre presente brindándome su gran apoyo y su compañía.

Diego Martín García Moya

AGRADECIMIENTOS

Agradezco a mis Padres por su apoyo incondicional en todo momento.

Agradezco a mi esposa e hijos por darme siempre esa inspiración y fortaleza para seguir adelante cada día.

Agradezco a la Universidad Politécnica Salesiana y a mis Maestros a lo largo de la carrera por transmitirme su conocimiento.

Agradezco a mi Tutor de Proyecto de titulación por su acompañamiento y guía para lograr el desarrollo de este trabajo.

Diego Martín García Moya

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

Resumen

El presente proyecto presenta una solución que apoyara a las actividades que se realizan en la veterinaria “Pets World” aportando mayor confiabilidad y rapidez al momento de gestionar la información referente a los pacientes, o al realizar despachos de las ventas realizadas en la petshop, también al requerir información actualizada de las existencias de productos y medicamentos en determinado momento, o cuando se necesita realizar un agendamiento de citas médicas. La solución propuesta permitirá incrementar el ahorro y buena utilización de material de oficina, eliminará la duplicación de historias clínicas, permitirá concretar ventas exitosamente, atenderá el abastecimiento de productos en bodega de manera eficiente y evitará el agendamiento incorrecto de citas médicas.

En la propuesta presentada se desarrolla un Sistema Web, el que principalmente servirá para gestionar toda la información generada en la veterinaria, permitiendo dar solución a los problemas de existencias de productos mediante monitoreo constante a los movimientos que se realizan con los productos en bodega y en la petshop, el sistema web muestra al usuario de manera rápida y sencilla toda la información referente a un paciente, permite agendar citas médicas mediante un calendario grafico que muestra los horarios y fechas disponibles y los no disponibles.

La implementación se la realiza usando la metodología Scrum, la cual permite, afrontar con éxito nuevos requerimientos del usuario, solventar posibles cambios en el entorno de la veterinaria

Con la implementación del sistema Web se ha logrado optimizar el tiempo empleado para la atención de un paciente, ahorrar en material de oficina, abastecer a la bodega de productos y medicamentos de manera oportuna y se conocen las fechas y horas disponibles y no disponibles al momento de agendar citas médicas.

Abstract

The present project presents a solution that supports the activities carried out in the veterinary "Pets World", providing greater reliability and speed when managing the information regarding patients, or when making shipments of sales made in the petshop, also by requiring updated information on the stock of products and medicines at a certain time, or when a schedule of medical appointments is needed. The proposed solution will increase the savings and good use of office equipment, eliminate duplication of medical records, will allow close sales successfully, will attend the supply of products in warehouse efficiently and prevent improper scheduling of medical appointments.

In the presented proposal a Web System is developed, which will mainly serve to manage all the information generated in the veterinary, allowing to solve the problems of stock of products through constant monitoring of the movements that are made with the products in the winery and in the petshop, the web system shows the user quickly and easily all the information regarding a patient, allows scheduling medical appointments through a graphic calendar that shows the schedules and dates available and those not available.

The implementation is done using the Scrum methodology, which allows successfully meet new user requirements, to solve possible changes in the veterinary environment

With the implementation of the Web system, it has been possible to optimize the time spent to care for a patient, save on office supplies, supply the warehouse with products and medicines in a timely manner and know the dates and times available and not available at the time. to schedule medical appointments.

INTRODUCCIÓN

El desarrollo de la tecnología ha permitido que la información se gestione de manera óptima, lo cual permite que los procesos que se realizan dependiendo de la información como base, lleguen a finalizarse correctamente, optimizando recursos y tiempo.

Las entidades comerciales se desenvuelven enmarcados en distintos giros de negocio, estas entidades para funcionar requieren de información de clientes, direcciones, etc., esta información, generada por las entidades comerciales tiende a crecer de manera exponencial, y en este punto es donde los sistemas de información se convierten en una parte importante para las entidades comerciales, ya que una de sus capacidades es recolectar o tomar la información que se genera en los departamentos o secciones de una entidad comercial, esta información es clasificada, organizada y almacenada de tal manera que posteriormente se la pueda disponer de manera que ayude en la toma de decisiones. La veterinaria "Pets World", ubicada en la ciudad de Quito, se dedica principalmente a brindar atención médica integral para animales domésticos y de granja, dentro y fuera de la ciudad, además comercializa productos, medicamentos y servicios como vacunación, peluquería, desparasitaciones, etc.

Todas las actividades de la veterinaria descritas anteriormente son llevadas o gestionadas de manera manual, llevando una agenda de citas en un cuaderno y el inventario de productos y medicamentos en una hoja de cálculo, lo cual genera algunos inconvenientes, los mismos que se explican a continuación.

Una de sus falencias está relacionada con el agendamiento de citas médicas para consulta, el problema está en que no se dispone de manera organizada las fechas y horas que ya han sido asignados a los pacientes, por lo que en reiteradas ocasiones se

han registrado citas para pacientes distintos en la misma fecha y hora, lo cual provoca la molestia de los clientes.

Un problema relacionado con las citas médicas agendadas es que al no tenerlas de manera organizada no se puede notificar a los clientes recordándoles que tienen una cita médica en una determinada fecha y hora, por tal motivo se han presentado olvidos por parte los clientes que finalmente no asistieron y perdieron la cita agendada.

Otro problema que tiene la veterinaria es al momento de vender productos o medicamentos, en varias ocasiones han tenido que anular ventas, porque no se puede despachar la cantidad de un pedido, ya que el stock que llevan de manera manual no coincide con la cantidad real de existencias en la bodega, y es en ese momento, cuando pierden una venta, que se realizan los ajustes en la hoja de cálculo para actualizar las existencias.

Otro inconveniente es con respecto al ingreso de información de pacientes, el problema radica en que, al no localizar la historia clínica de un paciente, se crea una nueva, con lo cual se desperdicia las hojas de formatos para historias clínicas, se duplica información, se prolonga el tiempo de atención para el paciente en un promedio de entre 10 y 15 minutos.

Para solucionar los problemas de la veterinaria “Pets World” se propone la creación de un sistema web para gestionar de manera apropiada los productos y servicios que se comercializan, este sistema web proporcionara un manejo adecuado de las existencias en bodega de los productos y medicamentos entregando las cantidades reales, las mismas que serán actualizadas de manera automática cuando haya ingresos o salidas de productos o medicamentos, con lo que se avala una respuesta adecuada ante un pedido.

Además, el sistema web permitirá agendar citas médicas de manera gráfica presentando un calendario en el cual se reflejen las fechas y horas que ya han sido asignadas con lo cual se solucionará el problema de agendamiento duplicado, con esto se incrementará la satisfacción de los clientes e imagen de eficiencia en la atención médica.

Otro beneficio del sistema web es que se podrá acceder a la información de los pacientes mediante búsquedas rápidas con lo cual se evitará la duplicación de información, ya no se utilizarán hojas de papel lo cual es un ahorro considerable al no gastar en hojas de formatos para historias clínicas y se disminuirá el tiempo empleado en una consulta en un promedio de entre 10 a 15 minutos.

También, el sistema web notificara mediante correos electrónicos cuando se haya alcanzado un stock mínimo para que la persona encargada realice los pedidos a los proveedores de manera oportuna.

Las notificaciones también serán enviadas mediante correo electrónico a los clientes de manera automática dos días antes de la fecha de la cita, con lo cual se solucionaría el problema de la no asistencia por olvido.

Objetivo General

Analizar, diseñar, desarrollar e implementar un Sistema Web que permita mejorar la gestión integral de productos y servicios, reducir el tiempo empleado en la atención medica veterinaria, informar sobre las existencias en bodega en tiempo real, prevenir la duplicidad en el agendamiento de citas y permitir el acceso rápido a la información.

Objetivos Específicos

Concretar ventas y despachar pedidos de manera exitosa mediante un control en tiempo real de las existencias en bodega de los productos y medicamentos que la veterinaria comercializa.

Evitar el agendamiento de citas duplicadas mediante un calendario virtual gráfico que permita registrar citas correctamente y alertando e impidiendo la duplicidad en la agenda.

Recordar de manera oportuna a los pacientes sobre una cita próxima, mediante el envío automático de un correo electrónico en el que se le notifique que debe asistir a una cita médica. De igual manera se notificará automáticamente a los encargados de adquisiciones cuando se alcance un valor de stock mínimo en bodega de alguno de los productos o medicamentos.

Reducir el tiempo empleado actualmente en la atención médica, mediante el acceso efectivo a las historias clínicas e información de pacientes.

CAPÍTULO 1

MARCO TEÓRICO

1.1 Scrum

Scrum es una metodología que gestiona y reduce la complejidad en el desarrollo de algún producto, esta metodología se fundamenta en el trabajo en equipo sobre pequeños bloques de trabajo que son parte de un proyecto grande, al seccionar el trabajo a realizar en partes pequeñas, scrum permite responde con mayor agilidad a posibles cambios o nuevos requerimientos a la vez que se inicia el desarrollo de la siguiente sección de trabajo.

"Scrum es un marco de trabajo para el desarrollo y el mantenimiento de productos complejos". (Ken Schwaber, 2017).

1.1.1 Historias de Usuario.

Es la materia prima de Scrum, todo empieza con los requerimientos y características que las personas involucradas tienen como perspectiva del software, las mismas son recogidas al detalle y se las llama historias de usuario, cada historia de usuario se la obtiene en torno a tres preguntas ¿Quién ejecutará la acción?, ¿Qué es lo que quiere hacer? Y ¿Para qué lo quiere hacer?

En la Guía de Scrum versión 2017 se describe a las historias de usuario como el componente principal del Product Backlog, "La Lista de Producto enumera todas las características, funcionalidades, requisitos, mejoras y correcciones que constituyen cambios a realizarse sobre el producto para entregas futuras" (Ken Schwaber, 2017, pág. 15).

1.1.2 Roles.

En un proyecto a desarrollarse se requiere de la participación de varias personas, estas personas ejecutan una o varias actividades, a las cuales se las conoce como Roles, en la metodología Scrum contamos con varios Roles los mismos que se describen a continuación:

1.1.2.1 Product Owner.

Como se mencionó con anterioridad el "Product Owner" es el encargado de la creación del "Product Backlog", registra las características adecuadas en el mismo, también es quien está en contacto permanente con el cliente, haciendo de intermediario entre el cliente y el equipo de desarrollo, de tal manera que es un aporte importante en el direccionamiento del producto. Schwaber y Sutherland (2017) dicen: "El Dueño de Producto es la única persona responsable de gestionar la Lista del Producto (Product Backlog)." (pág. 6).

1.1.2.2 Scrum Master.

También se lo puede considerar como el Administrador del Proyecto, es el encargado de que el proceso de desarrollo tenga un flujo suave, lo logra coordinando de manera eficiente las reuniones y realizando una óptima planeación de la liberación del producto "Release Planning".

"El Scrum Master es responsable de promover y apoyar Scrum como se define en la Guía de Scrum. Los Scrum Masters hacen esto ayudando a todos a entender la teoría, prácticas, reglas y valores de Scrum." (Ken Schwaber, 2017, pág. 8)

1.1.2.3 Equipo de Desarrollo.

El equipo de Desarrollo se divide en Developers y Testers, Los Developers son los que construyen el producto mediante la utilización de lenguajes de programación y otras

herramientas de desarrollo, los Testers son los encargados de probar funcionalidades, buscar errores, realizar todas las pruebas necesarias e informar a los desarrolladores.

"El Equipo de Desarrollo consiste en los profesionales que realizan el trabajo de entregar un Incremento de producto "Terminado" que potencialmente se pueda poner en producción al final de cada Sprint." (Schwaber, 2017, pág. 7)

1.1.3 Product Backlog.

Está constituido por las historias de usuario las mismas que representan al proyecto final, aquí se ordenan de manera tal que se les da un valor de prioridad, siendo las que tienen un valor más alto las que se ejecutaran primero, el responsable al cien por ciento de éste artefacto en el "Product Owner" ya que es el que está en contacto directo con el cliente.

1.1.4 Release Planning.

La Planificación de liberación de producto se realiza seleccionando las historias de usuario del Product Backlog que conformaran la lista de Release Planning, las historias de usuario de esta lista son priorizadas por el equipo, además se realiza una estimación de tiempo de culminación de cada componente y de esta manera se puede promediar el tiempo de desarrollo del producto final.

1.1.5 Sprint Backlog.

El Sprint Backlog se conforma de las historias priorizadas y estimadas del Release Planning, las historias priorizadas y estimadas se agrupan de tal manera que tengan relación entre si y que sus tiempos estimados de liberación no sean menores a 2 días ni mayores a 30 días, este espacio de tiempo se lo conoce como "Iteración", con cada iteración de sprints se realizan algunas etapas que son las siguientes:

1.1.5.1 Reunión de Planificación de Sprint.

En esta reunión se tratan básicamente dos aspectos que se pueden resumir en dos preguntas, ¿Qué se va a entregar en el próximo Sprint? y ¿Cómo se va a lograr concluir con éxito la tarea del próximo Sprint?, todo el equipo Scrum participa y colabora en la creación de la planificación.

1.1.5.2 Scrum Diario.

El Scrum Diario es un evento que se realiza diariamente con una duración de máximo quince minutos, tiempo en el cual brevemente el equipo de desarrollo coordina actividades para la siguiente jornada de trabajo, el beneficio de estas reuniones diarias es que se puede identificar posibles obstáculos o inconvenientes en el desarrollo, una ventaja es que mejora la comunicación del equipo y previene reuniones extra, y lo más importante permite monitorizar el avance del Sprint.

1.2 Sistema Web

Los Sistemas Web son aquellos que pueden ser utilizados o accedidos por medio de un navegador de internet, a través de una red de datos. Los sistemas Web son populares por la facilidad y practicidad que brindan para ser utilizados simplemente usando un navegador de internet como cliente, los navegadores son de fácil instalación, ligeros y finalmente no dependen de un sistema operativo como plataforma base.

1.2.1 Capas de los Sistemas Web.

La estructura de los Sistemas Web se basa en tres capas, cada una de estas cumple con una función específica, a continuación, se detalla cada una de las capas.

1.2.1.1 Capa del Navegador.

El navegador web es la capa intermedia entre el usuario y el sistema, el navegador solicita al servidor los archivos que conforman el sistema web, al obtenerlos interpreta

el código que puede ser HTML, CSS, XML, JavaScript y finalmente lo muestra en el navegador de internet para que el usuario inicie una interacción con el sistema.

1.2.1.2 Capa del Servidor.

En esta capa se aloja toda la codificación en lenguaje de programación web, también se encuentran todos las configuraciones y datos que permiten el funcionamiento del Sistema Web.

La siguiente tabla tomada de (TIOBE, 2019) muestra el porcentaje de uso de algunos lenguajes de programación Web en la actualidad.

Lenguajes de programación Web más usados.

Apr 2019	Apr 2018	Change	Programming Language	Ratings	Change
1	1		Java	15.035%	-0.74%
2	**2**		**C**	**14.076%**	**+0.49%**
3	3		C++	8.838%	+1.62%
4	4		Python	8.166%	+2.36%
5	6	^	Visual Basic .NET	5.795%	+0.85%
6	5	v	C#	3.515%	-1.75%
7	8	^	JavaScript	2.507%	-0.99%
8	9	^	SQL	2.272%	-0.38%
9	7	v	PHP	2.239%	-1.98%

Figura: 1 Contiene los porcentajes de uso de los lenguajes de programación
Fuente: tiobe.com

1.2.1.3 Capa de Persistencia.

Es la capa donde se encuentran las bases de datos donde se almacenan los datos e información con las que interactúa el sistema, mediante consultas a la base de datos se obtienen las vistas adecuadas, estas son traducidas a un lenguaje que pueda ser

interpretado por el navegador web y posteriormente se renderiza entregando la vista al usuario.

1.2.2 Backend.

El Backend es un término que hace referencia a una parte del sistema donde se encuentran los archivos con la codificación de acceso a bases de datos, algoritmos para cálculos, validaciones, etc. Un término relacionado con el Backend es "core" ya que es la sección del sistema donde se encuentra el motor que realiza el trabajo por detrás, de tal manera que es transparente al usuario. El Backend está relacionado con lenguajes como Php, Java, Python Ruby, .NET, etc.

1.2.3 Frontend.

El Frontend es la sección de sistema donde se alojan los archivos de lenguaje de programación y marcado que son interpretados por el navegador, el Frontend está relacionado con lenguajes como HTML, XML, JavaScript, CSS, entre otros. Es el intermediario entre el usuario y el Backend, pudiendo el usuario acceder directamente a las interfaces que ofrece el Frontend, los procesos tendrán efecto sobre el lado del cliente.

1.3 Patrón de arquitectura MVC

"Fue creada en 1979 por Trygve Reenskaug. Es un patrón que permite separar la GUI, de los datos y de la lógica apoyándose en tres componentes, Modelo, Vista y Controlador" (S.A.S, 2014, pág. 120).

Modelo, Vista y Controlador son los elementos que permiten separar la lógica del software de la interfaz, permite mejorar el mantenimiento y escalabilidad.

Ciclo de vida de la arquitectura MVC

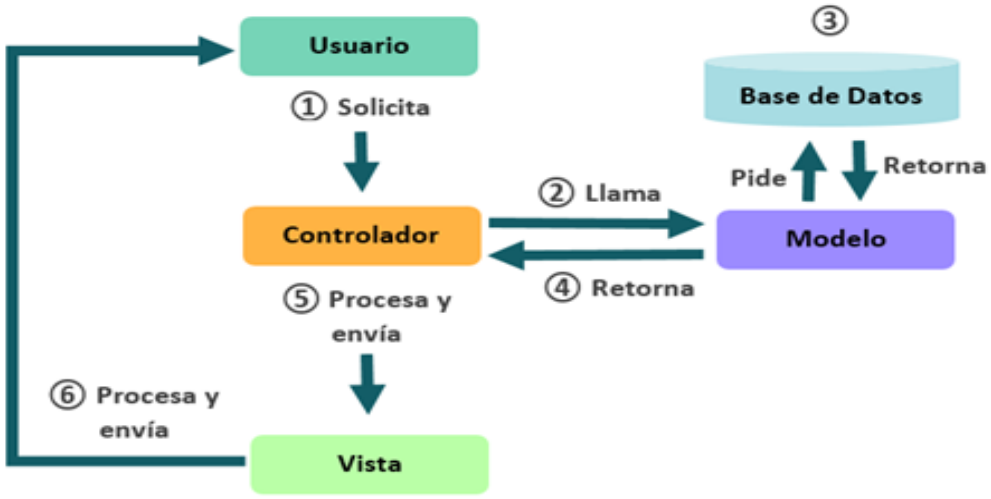


Figura: 2. Ciclo de vida del MVC

Fuente: (DevJoker, s.f.)

1.3.1 Definición de los componentes de la arquitectura MVC

1.3.1.1 Modelo.

En este componente se programa toda la interacciona con bases de datos, consultas, actualizaciones, ediciones y eliminaciones, es el encargado de manipular la información dependiendo del flujo del sistema, utiliza aquí persistencia de datos, de tal manera que las transacciones no sean afectadas.

1.3.1.2 Vista.

Este componente es el encargado de organizar la información que le entrega el controlador en la estructura de la vista de usuario, actualizando la información de manera automática en el caso de que cambie, esta organización y presentación de la información en el frontend es también conocida como renderizado.

1.3.1.3 Controlador.

El Controlador es el componente donde se localiza la programación para recibir las peticiones y órdenes del usuario, solicita data al modelo y los comunica a la vista para que los muestre. El Controlador es el intermediario entre la Vista y el Modelo, recibe, atiende y procesa las peticiones que le llegan.

1.3.2 Ciclo de vida de la arquitectura MVC.

- El usuario realiza una petición HTTP hacia el Controlador
- El controlador procesa dicha petición consultando la Base de Datos
- La Base de Datos responde con los datos solicitados y se los envía al Controlador
- El controlador los envía a la Vista para que los procese y organice en la estructura de la pagina
- La Vista renderiza la página en el navegador para mostrarla al usuario.

1.4 Lenguaje de programación PHP

Este lenguaje de programación fue creado en 1995 por Rasmus Lerdof, lo creó específicamente como un lenguaje para la web que posibilita la creación de sistemas dinámicos con acceso a información almacenada en bases de datos externas, sus siglas son un acrónimo recursivo de Hypertext Preprocessor o en español Procesador de Hipertexto, se distribuye a nivel mundial con licencia de tipo libre, Este lenguaje se ejecuta del lado del servidor, es invisible a los navegadores y al usuario, siendo además el primer lenguaje de programación web que se puede incrustar en las páginas HTML.

"En definitiva, PHP es uno de los lenguajes más utilizados actualmente en el desarrollo de aplicaciones Web y viene experimentando un constante crecimiento e su nivel de utilización en Internet." (Cobo, 2015, pág. 23)

1.5 Lenguaje de programación JavaScript

El lenguaje JavaScript también llamado JS es un dialecto del estándar ECMAScript lanzado en 1997, este leguaje es del tipo interpretado, ya que el navegador web es quien interpreta y ejecuta los scripts, JavaScript es además Orientado a Objetos lo cual le permite dotar de dinamismo al manipular el Document Object Model (DOM) de las páginas web, por este motivo JS es un lenguaje del lado del cliente, no obstante existe una versión que se ejecuta del lado del servidor, esta tecnología se denomina Node.Js.

JavaScript en sus versiones más recientes puede trabajar con tecnología AJAX lo que permite conectarse y obtener información de servidores externos sin necesidad de recargar la página.

“Como ya hemos dicho, JavaScript es un lenguaje universal presente en numerosas páginas HTML, de manera complementaria a este código.” (Gutierrez, 2009, pág. 13)

1.6 Motor de Base de Datos MySQL

“MySQL tiene como principal objetivo ser una base de datos fiable y eficiente.” (Rafael Camps Paré, 2015, pág. 256).

El motor de base de datos MySQL es un gestor de bases de datos relacional, utiliza SQL para la manipulación de los datos, MySQL es la distribución más extendida y usada en la comunidad informática ya que es de código abierto, es multiusuario y multiplataforma, permite el acceso a enlace de datos para manipularlos, su sintaxis es muy sencilla y fácil de aprender.

Una desventaja es que no es capaza de gestionar cantidades grandes de información, para lo cual se recomienda utilizar otras opciones más avanzadas. Es muy compatible

con el lenguaje PHP, existen herramientas que instalan en conjunto PHP y MySQL por la compatibilidad que estas dos tecnologías presentan.

1.7 Herramientas

1.7.1 LAMP.

Por sus siglas tenemos Linux, Apache, MySQL, PHP, este conjunto de herramientas conforman una infraestructura para la web, siendo Linux uno de los mejores sistemas operativos para configurar servidores que por naturaleza estarán expuestos a la internet, Linux brinda seguridad y niveles de acceso, Apache es un conocido servidor de Páginas Web, MySQL es un motor de base de datos relacional que presenta gran compatibilidad con el lenguaje Web PHP. LAMP instala todas las herramientas mencionadas en sus versiones más estables.

1.7.2 Sublime Text.

Es un editor de texto y de código fuente, que soporta gran cantidad de lenguajes de programación además se pueden incorporar plugins que permiten extender las funcionalidades del editor, está escrito en C++, no es software libre, sin embargo, su utilización no tiene fecha de vencimiento. Inicialmente fue creado como una extinción del editor Vim, este editor permite mostrar colores en las estructuras del lenguaje de programación lo cual facilita la lectura y escritura de código fuente, es ligero y mediante plugins permite la conexión con servidores externos para codificar directamente en los ficheros almacenados en ese recurso.

1.7.3 Node JS.

Node JS es un entorno de ejecución para JavaScrips construido con el motor de JavaScript V8 de Chrome (Node.JS, 2019)

Al estar programado con V8 se logra obtener independencia de los navegadores web, y se puede pasar a tener un comportamiento de servidor web que ofrece una funcionalidad específica, la que se desarrolla usando lenguaje de programación JavaScrtipt, permitiendo entre otras cosas crear API's, Streaming de audio o video, chats, etc. Todo esto porque ya se apoya como backend en el motor V8 el mismo que integra todas las características de ECMAScript en su versión más reciente.

Al estar construido orientado a eventos es más liviano y su estructura al trabajar con entradas y salidas sin bloqueos es más eficiente.

Node JS posee NPM que es el gestor de los paquetes y librerías para JavaScript, NPM trabaja de manera concomitante con WebPack para gestionar los recursos finales de los sistemas web.

1.7.4 WebPack.

Es una herramienta que permite generar los recursos finales que usara un Sistema web, se lo puede utilizar tanto para entornos de desarrollo como para entornos de producción, el funcionamiento básico es que toma nuestros archivos de configuración de JavaScript o CSS y los transpila en un solo archivo, estos assets transpilados permiten que el sistema sea soportado con la mayoría de navegadores web existentes.

Para su funcionamiento es necesario la instalación de Node previamente, entre las ventajas tenemos que el uso de Webpack mejora la velocidad de carga ya que además optimiza las imágenes reduciendo su peso original sin afectar a la calidad, también brinda facilidad en la administración en un único fichero público.

1.7.5 Frameworks.

Los Frameworks son esquemas de trabajo o marco de trabajo que ofrece un esqueleto en el cual armar una arquitectura que dará vida a un determinado Proyecto, el modelo

MVC (Modelo, Vista, Controlador) es muy utilizado por los marcos de trabajo para ayudar al desarrollo de las aplicaciones, a continuación, se describen los Frameworks usados en el presente trabajo de titulación.

1.7.5.1 Laravel.

Laravel es un Framework para el lenguaje de programación PHP, posibilita la creación de código que se basan en el modelo MVC (Modelo, Vista, Controlador). Este Framework cuenta con una extendida fuente de documentación la misma que es mantenida por sus desarrolladores y por una comunidad de programadores alrededor del mundo. Sus características se detallan a continuación:

1.7.5.1.1 Blade .

Es el motor de plantillas de Laravel que facilita la creación de Vistas, permite extender plantillas o incrustar secciones en otras Vistas, también permite el uso de PHP en las mismas.

1.7.5.1.2 Eloquent.

Es el ORM de Laravel que gestiona toda la interacción con las Bases de Datos, su funcionamiento consiste en transformar las peticiones SQL en MVC para proporcionar seguridad de tal manera se pueda evitar ataques de inyección SQL

1.7.5.1.3 Middlewares.

Son secciones de código que funcionan como intermediarias entre las peticiones al servidor, esto posibilita que aquí se incrusten validaciones, procesos alternos, etc. Ya que al encontrarse en el medio del flujo de una petición se puede actuar sobre los datos que viajan en dicha petición, incrementando así la seguridad de las aplicaciones.

1.7.5.1.4 Commands.

Laravel posee comandos que realizan acciones como por ejemplo realizar respaldos de información, guardar archivos, etc. El Framework Laravel permite crear comandos personalizados que se ajusten a los requerimientos del desarrollador, de esta manera se pueden crear comandos que realicen de manera automática el envío de emails de felicitación a clientes que estén próximos a cumplir años.

1.7.5.2 Bootstrap.

Es un marco de trabajo de código abierto, es independiente de la plataforma y esta creado para el trabajo en el Frontend administrando una gran variedad de componentes HTML o CSS, además en sus últimas versiones se puede encontrar extensiones a librerías JavaScript, todo este conjunto de características permite que el Framework sea compatible con la mayoría de navegadores web.

1.7.5.3 VueJS.

Este es un marco de trabajo que fue diseñado para trabajar sobre el Frontend, el núcleo es muy liviano y puede ser escalable mediante plugins, al poder el desarrollador configurar su framework con lo necesario para conseguir algún Proyecto se logra velocidad. Mediante el uso de este framework se pueden crear aplicaciones de una sola página como los SPA

Es reactivo ya que se permiten realizar una acción por cada vez que se activa un evento del sistema que puede ser interno o externo, es decir reacciona al cambio y está en constante conexión con el entorno del sistema.

Es muy recomendable usarlo en sistemas en los que se manejaran gran cantidad de eventos que pueden ser generados por usuarios, otros sistemas o por sensores electrónicos.

1.7.5.4 JQuery.

JQuery es considerado la mejor biblioteca de JavaScript, permite al desarrollador emplear menos tiempo de desarrollo que lo que emplearía en una aplicación echa con JavaScript puro, también hace más livianas las aplicaciones ya que usan menos código para lograr la misma funcionalidad.

Su funcionalidad se basa en la manipulación de todos los elementos del DOM, esto implica también la parte de diseño y estilos CSS y finalmente solicitando peticiones AJAX, estas son las características más relevantes,

Posee una documentación muy extendida y además está en constante mantenimiento, esta tecnología es base para algunas de las funcionalidades de Bootstrap en sus últimas versiones.

1.7.6 Balsamiq Mockups.

Es una herramienta que permite a los desarrolladores crear bocetos o prototipos de las vistas que se crearan para conformar la GUI, actualmente posee una versión de uso gratuito, multiplataforma y una versión Web. Es una herramienta indispensable para el desarrollo de todo tipo de aplicaciones que requieren llevar una documentación detallada de su construcción.

1.7.7 Power Designer.

Power Designer es una herramienta de modelado de colaboración empresarial creada por la Sybase, permite la creación y diseño de infinidad de diagramas y modelos, pasando por diagramas UML, Modelado de BDD y otros.

1.7.8 Composer.

Composer es un manejador de Dependencias creado exclusivamente para PHP, es el encargado de descargar todas las librerías y paquetes en las versiones específicas que

un sistema Web echo con PHP puede necesitar, si las dependencias a su vez tienen otras dependencias creadas por terceros, Composer resuelve ese inconveniente y realiza descargas completamente transparentes al desarrollador, lo cual lo convierte en algo de muchísima utilidad, ya que antes de su existencia para obtener paquetes o librerías se tenía que ir a la página del creador que en muchos casos no la tenía, estas páginas requerían de registro de usuario, todo esto implicaba tiempo valioso de desarrollo, con Conocer lo engorroso de descargar dependencia y paquetes se convierte en algo muy sencillo.

1.7.9 Axios.

Es un cliente que trabaja usando el protocolo HTTP para realizar peticiones de tipo XMLHttpRequst, las mismas que son emitidas desde una aplicación JavaScript que puede estar escrita en lenguaje puro o mediante algún framework. Axios puede trabajar en el lado del cliente y en el lado del servidor.

1.7.10 Cron.

Es una aplicación presente en sistemas operativos Linux, su función es la de ejecutar un demonio que analiza el contenido en un archivo llamado “Crontab” y realiza las actividades y acciones allí descritas de manera periódica, por ejemplo, cada hora, cada, día, cada mes, etc. Las actividades en el archivo “Crontab” se escriben usando sintaxis propia de “Cron”.

CAPÍTULO 2

ANALISIS Y DISEÑO

2.1 Análisis de Requerimientos

Mediante varias reuniones con el representante de la institución se ha logrado obtener los requerimientos del software, además se han recolectado documentación física que se genera producto del funcionamiento cotidiano de la entidad, a partir de toda esta información se establecen los siguientes requerimientos funcionales y no funcionales.

2.1.1 Requerimientos Funcionales.

A continuación, se enlistan las funcionalidades que el usuario necesita que el sistema pueda hacer.

2.1.1.1 Requerimiento Funcional 1: Administración de Productos.

Se requiere mantener control de las existencias en bodega, de manera que cuando un determinado producto alcance un valor de stock mínimo, el sistema de manera automática envíe un email a la persona encargada de bodega.

Este requerimiento al ser administrativo, se requiere de las funcionalidades de agregar producto, editar producto, eliminar producto, y ver producto, las mismas que se describen a continuación.

Tabla 1. Requerimiento Funcional 1: Agregar Producto

Descripción	Agregar Producto
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.

Entrada	Información detallada del producto: nombre, descripción categoría, precio de venta, costo, impuesto, estado, stock, stock mínimo.
Proceso	Verificación de credenciales del usuario Ingresar al módulo de Productos Llenar la información del producto Validación de campos Registro en la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 1 (agregar producto)

La tabla 1, describe la funcionalidad de agregar producto, que la podrán ejecutar los usuarios que dispongan de los roles y permisos adecuados, proporcionando información válida para procesarla y finalmente recibir una confirmación.

Tabla 2. Requerimiento Funcional 1: Editar Producto

Descripción	Editar Producto
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Datos para modificar la información del producto
Proceso	Verificación de credenciales del usuario Ingresar al módulo de Productos, Cambiar la información del producto Validación de campos Actualización de campos en la Base de Datos

Salida	Mensaje de error a confirmación, además se envía un email dependiendo el stock actual del producto.

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 1 (editar producto)

La tabla 2, muestra la funcionalidad de editar producto, esta función será accesible solamente para usuarios verificados, los mismos que al editar la información existente se validará y procesará adecuadamente, finalmente se mostrará un mensaje de error o confirmación.

Tabla 3. Requerimiento Funcional 1: Eliminar Producto

Descripción	Eliminar Producto
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	El código del producto a eliminar
Proceso	Verificación de credenciales del usuario Ingresar al módulo de Productos, Seleccionar el producto a eliminar Validación del código del producto Eliminación del registro de la Base de Datos Verificación de stock mínimo y el stock existente
Salida	Mensaje de error a confirmación.

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 1 (eliminar producto)

La tabla 3, contiene la descripción de la funcionalidad de eliminar producto, la misma que podrá ser utilizada por usuarios con permisos adecuados, al realizar la petición de eliminar un registro se validará el proceso y mostrará un mensaje de confirmación o error según el caso.

Tabla 4. Requerimiento Funcional 1: Ver Producto)

Descripción	Ver Producto
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Código del producto a visualizar
Proceso	Verificación de credenciales del usuario Ingresar al módulo de Productos Seleccionar el producto a visualizar Validación del código del producto Consulta a la Base de Datos
Salida	Ventana con la información completa del producto

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 1 (ver producto)

La tabla 4, describe la funcionalidad cuando usuarios con permisos adecuados solicite listar y ver los productos existentes, posteriormente se emitirá una respuesta de error o confirmación y la vista de usuario adecuada.

2.1.1.2 Requerimiento Funcional 2: Administración de Categorías

Se requiere que el sistema permita administrar categorías, las mismas que serán asignadas posteriormente a los productos, cada categoría se podrá editar, visualizar, o eliminar.

Tabla 5. Requerimiento Funcional 2: Agregar Categoría)

Descripción	Agregar Categoría
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Nombre, descripción, estado

Proceso	Verificación de credenciales del usuario Ingresar al módulo de Categorías Llenar el formulario con los datos de la categoría Validación de campos Registro en la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 2 (agregar categoría)

La tabla 5, describe la funcionalidad de agregar categoría, la misma que podrá ser utilizada por usuarios con roles adecuados, estos usuarios ingresan la información que será validada para posteriormente emitir un mensaje de error o confirmación

Tabla 6. Requerimiento Funcional 2: Editar Categoría

Descripción	Editar Categoría
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Código de la categoría a editar Información para editar los campos de la categoría
Proceso	Verificación de credenciales del usuario Ingresar al módulo de Categorías Llenar el formulario con los datos de la categoría Validación de campos Edición del registro en la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 2 (editar categoría)

En la tabla 6, se describe la función de editar categorías, la podrán realizar usuarios verificados únicamente, la información proporcionada será validada y según la respuesta se emitirá un mensaje para el usuario.

Tabla 7. Requerimiento Funcional 2: Eliminar Categoría

Descripción	Eliminar Categoría
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Código de la categoría a eliminar
Proceso	Verificación de credenciales del usuario Ingresar al módulo de Categorías Seleccionar la categoría a eliminar Validación del código de la categoría Eliminación del registro en la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 2 (eliminar categoría)

La tabla 7, describe el requerimiento eliminar categoría, este requerimiento será ejecutado por usuarios con permisos adecuados, se validará la información proporcionada por el usuario y se emitirá un mensaje al usuario dependiendo el caso.

Tabla 8. Requerimiento Funcional 2: Ver Categoría

Descripción	Ver Categoría
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Código del producto a visualizar

Proceso	Verificación de credenciales del usuario Ingresar al módulo de Categorías Seleccionar la categoría a visualizar Validación del código de la categoría Consulta a la Base de Datos
Salida	Ventana con la información completa de la categoría

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 2 (ver categoría)

La tabla 8, describe el requerimiento ver categoría, los usuarios con permisos adecuados podrá ejecutar esta función, se validará la información que el usuario entrega y se mostrará un mensaje e confirmación o error y además la vista adecuada.

2.1.1.3 Requerimiento Funcional 3: Envió de Notificaciones vía email.

Se requiere que el sistema envié una notificación automática a usuarios del sistema cuando se agoten productos en bodega y también a clientes, cuando se aproxime la fecha de una cita agendada.

Tabla 9. Requerimiento Funcional 3: Notificaciones para Bodega

Descripción	Notificaciones vía email para encarado de bodega
Precondición	La actualización del campo stock en la base de datos de algún producto específico.
Entrada	Código del producto que se encuentra con existencias bajas
Proceso	Se crea el email con los datos del usuario destinatario y los datos del detalle del producto, especificando las existencias actuales y las existencias mínimas
Salida	Mensaje de error o confirmación, un email al destinatario adecuado.

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 3 (agregar producto)

La tabla 9, describe el requerimiento notificaciones para bodega, este debería ejecutarse cada vez que se realice un movimiento con algún producto, por ejemplo una venta o una compra, al alcanzar o estar por debajo de un valor de existencias pre establecido se enviara un email al encargado de bodega.

Tabla 10. Requerimiento Funcional 3: Notificaciones para clientes

Descripción	Notificaciones vía email para clientes.
Precondición	Que la fecha de una cita agendada se alcance en 24 horas.
Entrada	Código del cliente para el envío de la notificación.
Proceso	Se crea el email con los datos del cliente y los datos del detalle de la cita agendada, especificando fecha y hora en la que se realizara.
Salida	Mensaje de error o confirmación, un email al destinatario adecuado.

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 3 (para clientes)

La tabla 10, describe el requerimiento notificaciones para clientes el cual será ejecutado por el sistema operativo de manera automática diariamente, se realizará una consulta de las citas médicas agendadas para el día siguiente, y se notificará vía email a cada uno de los clientes que tengan una cita en las siguiente 24 horas.

2.1.1.4 Requerimiento Funcional 4: Administración de Clientes.

Se requiere llevar de manera detallada la información de los clientes, esta información se relacionará con los pacientes, se necesita realizar búsquedas por nombre o cédula de identidad del cliente para la administración de los mismos, se requiere crear, editar, eliminar o ver la información de los clientes.

Tabla 11. Requerimiento Funcional 4: Agregar Cliente

Descripción	Agregar Cliente
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Cédula, nombre, apellidos, teléfonos, email, dirección, sector
Proceso	Verificación de credenciales del usuario Ingresar al módulo de Clientes Llenar el formulario con los datos del cliente Validación de campos Registro en la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 4 (agregar cliente)

En la tabla 11, se describe la funcionalidad al agregar un Cliente, un usuario con permisos de agregar cliente ingresa los datos nuevos que serán validados y posteriormente procesados, finalmente recibe un mensaje de confirmación.

Tabla 12. Requerimiento Funcional 4: Editar Cliente

Descripción	Editar Cliente
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Datos con los que será editado el registro de determinado cliente
Proceso	Verificación de credenciales del usuario Ingresar al módulo de Clientes Cambiar el formulario con los datos del cliente Validación de campos Edición del registro en la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 4 (editar cliente)

La tabla 12, describe la funcionalidad de editar cliente, que podrá ser ejecutada únicamente por usuarios con permisos adecuados, los datos a editar son validados y procesados para entregar dependiendo el caso un mensaje de confirmación o error.

Tabla 13. Requerimiento Funcional 4: Eliminar Cliente

Descripción	Eliminar Cliente
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Código del cliente a eliminar
Proceso	Verificación de credenciales del usuario Ingresar al módulo de Clientes Validación de código Eliminación del registro en la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 4 (eliminar cliente)

En la tabla 13, podemos observar la funcionalidad de eliminar cliente, aquí se debe validar los permisos del usuario que ejecuta la función y el id del cliente que será eliminado, se recibirá un mensaje de error o confirmación.

Tabla 14. Requerimiento Funcional 4: Ver Cliente

Descripción	Ver Cliente
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Código del cliente a visualizar
Proceso	Verificación de credenciales del usuario Ingresar al módulo de Clientes Seleccionar el cliente a visualizar

	Validación del código Consulta a la Base de Datos
Salida	Ventana con toda la información de un cliente especifico

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 4 (ver cliente)

La tabla 14, describe la funcionalidad de ver cliente que será ejecutada por usuarios con los permisos adecuados, como respuesta se muestra una interfaz con un listado e todos los clientes registrados en el sistema.

2.1.1.5 Requerimiento Funcional 5 Administración de Pacientes.

Se requiere que el sistema administre el historial de un paciente, permitiendo manipular, eliminar, visualizar toda la información de un determinado paciente.

Tabla 15. Requerimiento Funcional 5: Agregar Paciente

Descripción	Agregar Paciente
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Código del propietario, nombre, especie, color, sexo, fotografía, peso, fecha de nacimiento, chip, señas particulares
Proceso	Verificación de credenciales del usuario Ingresar al módulo de Pacientes Llenar el formulario con los datos del paciente Validación de campos Registro en la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 5 (agregar paciente)

La tabla 15, contienen la funcionalidad agregar paciente del requerimiento funcional 5, Administrar Pacientes, esta será accesible solo para usuarios con permisos

adecuados, los daos son validados y procesados para finalmente entregar una retroalimentación al usuario como un mensaje de confirmación o error dependiendo el caso.

Tabla 16. Requerimiento Funcional 5: Editar Paciente

Descripción	Editar Paciente
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Datos con los que será editado el registro de determinado paciente
Proceso	Verificación de credenciales del usuario Ingresar al módulo de Pacientes Cambiar el formulario con los datos del paciente Validación de campos Edición del registro en la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 5 (editar paciente)

En la tabla 16, se describe la funcionalidad de editar paciente, un usuario con permisos necesarios modificará la información de determinado paciente, esta información será validad y procesada, posteriormente se enviará un mensaje al usuario.

Tabla 17. Requerimiento Funcional 5: Eliminar Paciente

Descripción	Eliminar Paciente
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Código del paciente a eliminar
Proceso	Verificación de credenciales del usuario Ingresar al módulo de Pacientes Seleccionar el cliente a eliminar

	Validación del código Eliminación del registro en la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 5 (eliminar paciente)

La tabla 17, contienen la descripción de la funcionalidad eliminar paciente, esta podrá ser ejecutada únicamente por usuarios con los permisos y roles adecuados, el id del paciente y los roles de usuario serán validados y procesados, entregando finalmente un mensaje al usuario.

Tabla 18. Requerimiento Funcional 5: Ver Paciente

Descripción	Ver Paciente
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Código del paciente a visualizar
Proceso	Verificación de credenciales del usuario Ingresar al módulo de Pacientes Seleccionar el cliente a visualizar Validación del código Consulta a la Base de Datos
Salida	Ventana con toda la información del paciente

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 5 (ver paciente)

La tabla 18, describe la funcionalidad de ver paciente, que será ejecutada por usuarios con permisos adecuaos, como respuesta a la petición se mostrará una vista de interfaz de usuario con una tabla del listado de pacientes registrados en el sistema.

2.1.1.6 Requerimiento Funcional 6 Facturación.

Facturación: Se requiere que el sistema maneje un control de la facturación realizada, imprima dichas facturas para ser entregadas a los clientes, por compra de productos o servicios.

Tabla 19. Requerimiento Funcional 6: Listado de facturas pendientes de cobro

Descripción	Ver facturas pendientes de cobro
Precondición	Se debe haber realizado una atención en veterinaria o una venta de productos.
Entrada	Códigos de las facturas pendientes de cobro
Proceso	Verificación de credenciales del usuario Ingresar al módulo de Facturas Consulta a la Base de Datos
Salida	Ventana con toda la información sobre facturas pendientes

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 6 (ver facturas)

La tabla 19, describe la funcionalidad de listar facturas pendientes de cobro, cuando un usuario con permisos adecuados solicita listar las facturas con estado pendiente de cobro, se validaran las credenciales del usuario y se presentara la interfaz con la información solicitada por el usuario.

Tabla 20. Requerimiento Funcional 6: Cobrar factura pendiente

Descripción	Cobrar factura pendiente
Precondición	Se debe haber realizado una atención en veterinaria o una venta de productos.
Entrada	Códigos de la factura
Proceso	Verificación de credenciales del usuario Ingresar al módulo de Facturas

	Seleccionar la factura a cobrar Registrar cobro en la Base de Datos Actualizar el estado de la factura Imprimir factura y entregar al cliente
Salida	Mensaje de error o confirmación y un PDF

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 6 (cobrar factura pendiente)

En la tabla 20, se describe la funcionalidad esperada cuando un usuario con permisos adecuados realice el cobro de una factura, se construirá la vista para ser mostrada al usuario con el detalle de la factura a cobrar y generara un pdf para entregar al cliente.

2.1.1.7 Requerimiento Funcional 7 Generación de Reportes.

Se requiere que el sistema pueda generar reportes de recetas médicas y tratamientos en formato de documento portable, estos reportes serán entregados al cliente.

Tabla 21. Requerimiento Funcional 7: Generación de Receta Médica

Descripción	Generación de Receta Médica
Precondición	Se debe realizar una atención médica
Entrada	Códigos de la consulta
Proceso	Creación de la receta médica Imprimir receta y entregar al cliente
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 7 (generación de receta médica)

La tabla 21, describe la funcionalidad de generar receta médica, esta funcionalidad se espera cuando un médico realiza una atención médica, se imprimirá un pdf con el detalle de la receta médica y sus indicaciones.

Tabla 22. Requerimiento Funcional 7: Generación Tratamiento

Descripción	Generación de Tratamiento
Precondición	Se debe realizar una atención médica
Entrada	Códigos de la consulta
Proceso	Creación del tratamiento Imprimir tratamiento y entregar al cliente
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 7 (generación de Tratamiento)

Como se puede ver en la tabla 22, se describe la funcionalidad esperada cuando un médico concluya una cita médica, se imprimirá un pdf que contienen el tratamiento para ser entregado al cliente.

2.1.1.8 Requerimiento Funcional 8 Agendamiento de Citas.

El sistema administrará el agendamiento de nuevas citas para los pacientes, además enviará notificaciones vía email recordando a los clientes que una cita médica esta próxima.

Tabla 23. Requerimiento Funcional 8: Agendar Cita Médica

Descripción	Agendar Cita Medica
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Código del paciente
Proceso	Ingresar al módulo agendar Citas Verificar el código del paciente Creación de la cita médica en la agenda Enviar notificación vía email sobre la nueva cita agendada
Salida	Mensaje de error o confirmación, notificación email al cliente

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 8 (agendar Cita Médica)

La tabla 23, describe la funcionalidad requerida al momento de que un usuario con permisos adecuados realiza el agendamiento de una cita médica, se debe verificar horarios y fechas disponibles y ocupadas, y registrar en la base la cita médica, también se requiere el envío de una notificación de la cita médica con 24 horas de antelación.

2.1.1.9 Requerimiento Funcional 9 Sistema Multiusuario.

Se requiere que el sistema permita usuarios con roles específicos y acceso a determinadas partes del sistema, entre estos usuarios tenemos: médico, vendedor, cajero, bodeguero, secretaria.

Tabla 24. Requerimiento Funcional 9: Crear Usuario

Descripción	Crear Usuario
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Nombre, email, contraseña
Proceso	Ingresar al módulo Usuarios Presionar en el botón Crear Usuario Llenar el formulario con los datos solicitados Validar los campos del formulario Registrar el nuevo usuario en la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 9 (crear usuario)

La tabla 24, muestra el requerimiento al momento de que un usuario con roles de administrador cree un usuario, se envía los datos del usuario para ser validados y procesados en la base de daros, finalmente dependiendo del caso se muestra un mensaje de error o confirmación.

Tabla 25. Requerimiento Funcional 9: Editar Usuario

Descripción	Editar Usuario
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Datos con los que se editara al usuario, código del usuario a editar
Proceso	Ingresar al módulo Usuarios Seleccionar al usuario a editar Editar la información anterior Validar los campos del formulario Registrar los cambios en la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 9 (editar usuario)

En la tabla 25, se describe la funcionalidad esperada cuando un usuario con permios adecuados edite la información de un usuario creado, se validará la información a editar y se procesara, posteriormente se debe mostrar un mensaje al usuario.

Tabla 26. Requerimiento Funcional 9: Eliminar Usuario

Descripción	Eliminar Usuario
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Código del usuario a eliminar
Proceso	Ingresar al módulo Usuarios Seleccionar al usuario a eliminar Eliminar registrar de la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 9 (eliminar usuario)

La tabla 26, muestra la funcionalidad esperada cuando se requiera eliminar un usuario, se validará la identificación del usuario a eliminar y se procesara la petición, además se requiere mostrar un mensaje al usuario.

Tabla 27. Requerimiento Funcional 9: Ver Usuario

Descripción	Ver Usuario
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Código del usuario a visualizar
Proceso	Ingresar al módulo Usuarios Seleccionar al usuario a visualizar Consultar la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 9 (ver usuario)

La tabla 27, describe el requerimiento de ver usuario, esta funcionalidad la realizara un usuario con permisos adecuados, los daros de entrada deberán ser procesados y validados para después mostrar un mensaje de confirmación y la vista adecuada al usuario.

Tabla 28. Requerimiento Funcional 9: Crear Rol

Descripción	Crear Rol
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Nombre, url, descripción, permiso especial, permisos asignados
Proceso	Ingresar al módulo Roles Presionar en el botón Crear Rol Llenar el formulario con los datos solicitados Validar los campos del formulario

	Registrar el nuevo rol en la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 9 (crear rol)

La tabla 28, define el requerimiento de crear un rol, el mismo que debe ser ejecutado por un usuario con roles adecuados, el mismo proporcionara la información del nuevo rol y la enviara para que sea validad y procesada, al finalizar se mostrara un mensaje informativo al usuario.

Tabla 29. Requerimiento Funcional 9: Editar Rol

Descripción	Editar Rol
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Código del rol a editar, datos con los que se editara loa información del rol
Proceso	Ingresar al módulo Roles Seleccionar el Rol a editar Llenar el formulario con los datos Validar los campos del formulario Registrar las modificaciones del rol en la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 9 (editar rol)

La tabla 29, contiene la funcionalidad esperada cuando un usuario requiera editar la información de un rol, los datos a editar se validan y de estar todo correcto se requiere guardar los cambios en la base de datos.

Tabla 30. Requerimiento Funcional 9: Eliminar Rol

Descripción	Eliminar Rol
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Código del rol a eliminar
Proceso	Ingresar al módulo Roles Seleccionar el Rol a eliminar Eliminar el rol de la Base de Datos
Salida	Mensaje de error o confirmación

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 9 (eliminar rol)

La tabla 30, muestra el requisito cuando un usuario validado desee eliminar un rol, la información del rol a editar es verificada y se procesa mostrando un mensaje de información al usuario.

Tabla 31. Requerimiento Funcional 9: Ver Rol

Descripción	Ver Rol
Precondición	Usuario correctamente verificado en el sistema, roles y permisos de usuario con capacidad para ejecutar la funcionalidad.
Entrada	Código del rol a visualizar
Proceso	Ingresar al módulo Roles Seleccionar el Rol a visualizar Consultar la Base de Datos
Salida	Ventana con toda la información del Rol seleccionado.

Nota: Esta tabla contiene la especificación del Requerimiento Funcional 9 (ver rol)

La tabla 31 describe el requerimiento de ver un rol, esta funcionalidad la usará un usuario validado que al enviar la petición se construirá una interfaz para mostrar al usuario la información del rol solicitado.

2.1.2 Requerimientos no funcionales.

Los requerimientos no funcionales están conformados por las restricciones que se describen a continuación.

- El sistema tiene que ser accesible desde internet con el uso de un navegador web.
- La interfaz gráfica debe ser limpia y de fácil lectura e interpretación.
- Se deben utilizar para el desarrollo del sistema el framework Laravel para el backend y Vue JS para el frontend.

2.2 Modelado del Sistema

2.2.1 Diagramas de Casos de Uso.

Es una descripción grafica de la interacción de los usuarios con el sistema en una tarea específica, representan las acciones que los actores pueden realizar y los módulos del sistema que participan en determinada interacción. Aquí se presentan los casos de uso del sistema.

2.2.1.1 Caso de uso "Funcionalidad General".

A continuación, se presenta el diagrama de casos de uso general, en el que se muestran los actores y su relación con los casos de uso, los actores tienen acceso especifico a determinados módulos del sistema de tal manera se muestra a continuación.

Caso de Uso: Funcionalidades Generales

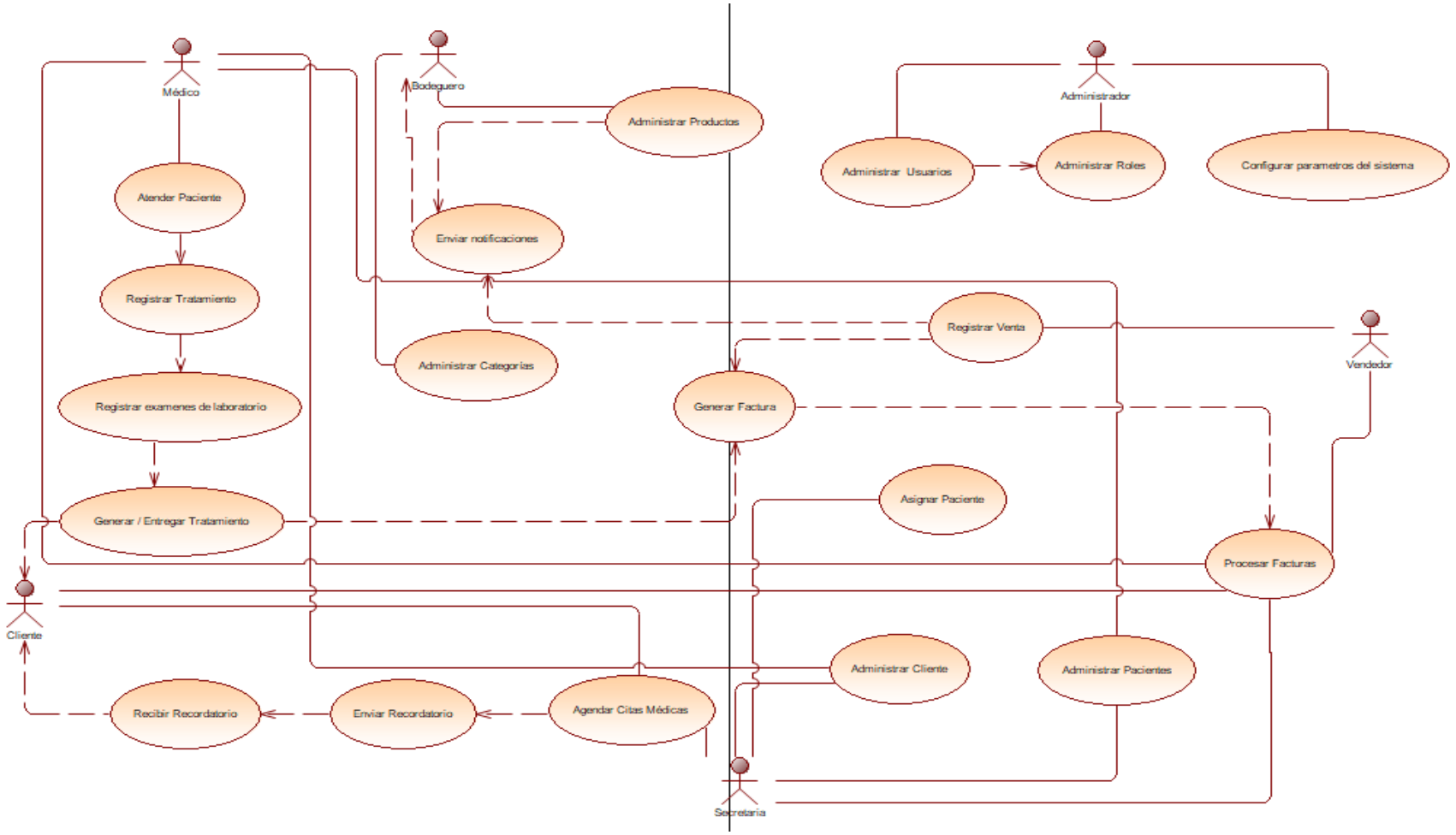


Figura: 3. Diagrama de Casos de Usos del sistema en General.
Elaborado por: Diego García Moya

2.2.1.2 Casos de Uso Funcionalidad módulo Productos.

Mediante el diagrama de la figura 4, se representa todas las interacciones del actor "bodeguero" con los casos de uso para el módulo de Administración de Productos, este actor posee permisos totales sobre este módulo pudiendo realizar acciones como crear, leer, actualizar, eliminar, filtrar.

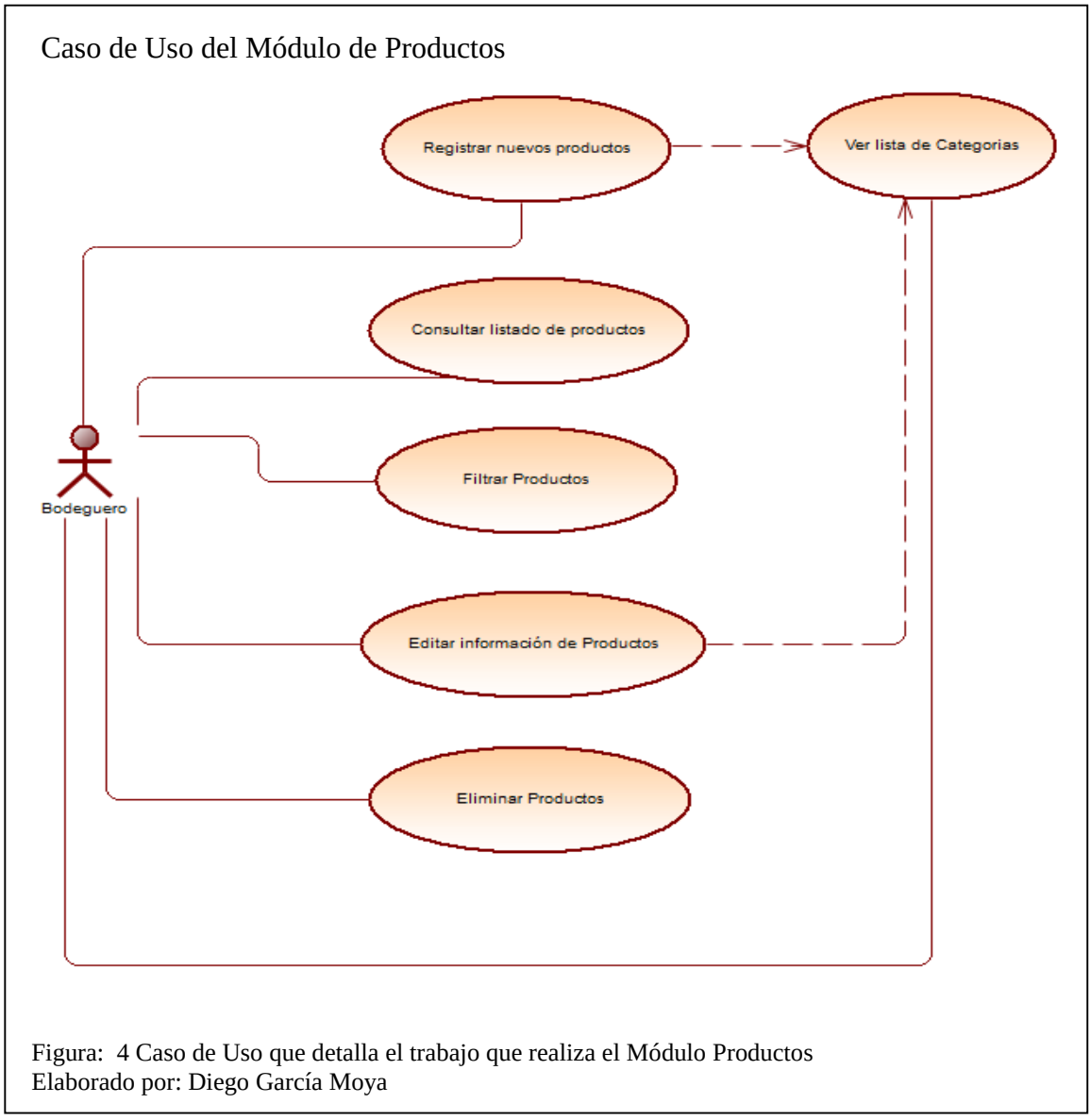


Figura: 4 Caso de Uso que detalla el trabajo que realiza el Módulo Productos
Elaborado por: Diego García Moya

2.2.1.3 Caso de Uso Funcionalidad módulo Categorías.

Mediante el diagrama de la figura 5, se representa todas las interacciones del actor "bodeguero" con los casos de uso para el módulo de Administración de Categorías, este actor posee permisos totales sobre este módulo pudiendo realizar acciones como crear, leer, actualizar, eliminar.

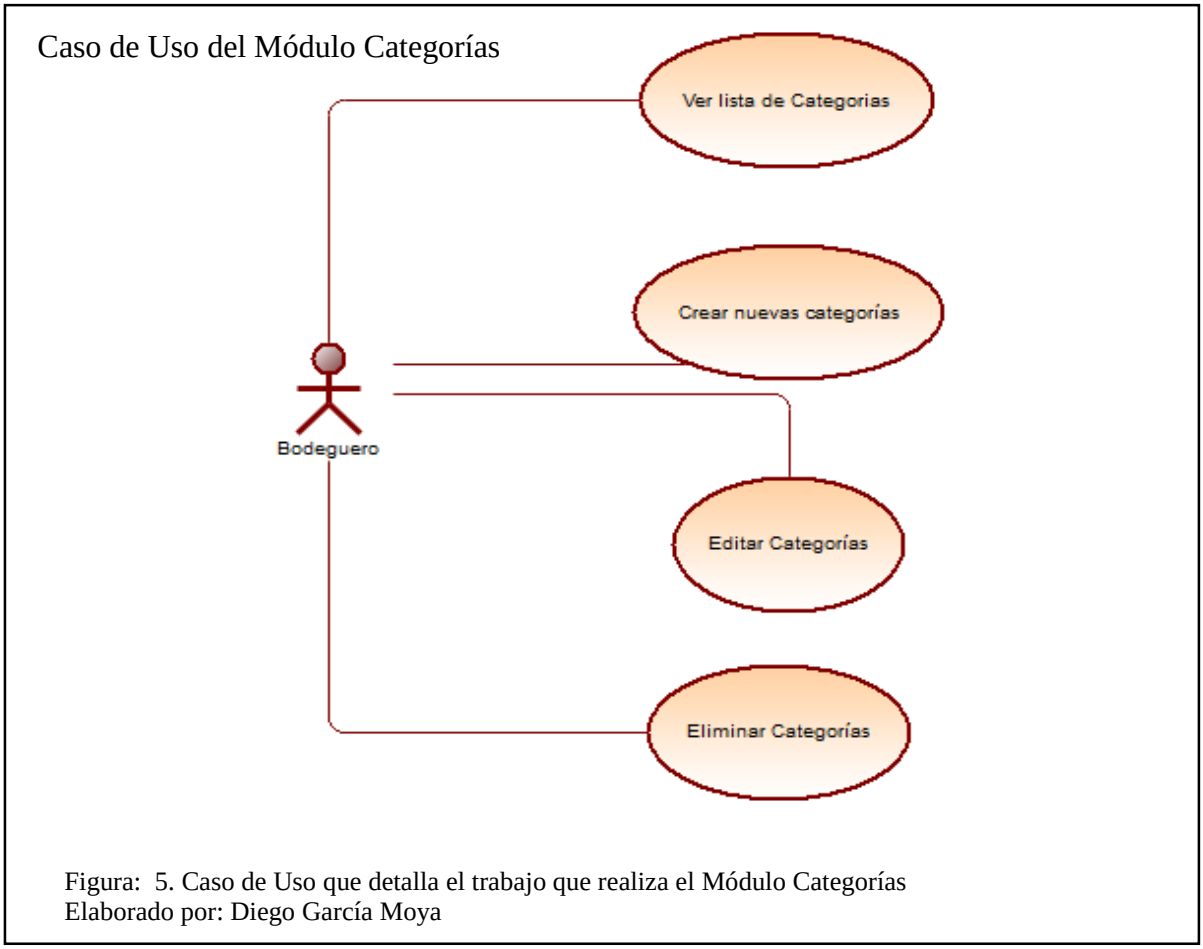


Figura: 5. Caso de Uso que detalla el trabajo que realiza el Módulo Categorías
Elaborado por: Diego García Moya

2.2.1.4 Caso de Uso Funcionalidad módulo Clientes.

Mediante el diagrama de la figura 6, se representa todas las interacciones de los actores "Secretaria" y "Médico" con los casos de uso para el módulo de Administración de Clientes, estos actores poseen permisos totales sobre este módulo pudiendo realizar acciones como crear, leer, actualizar, eliminar.

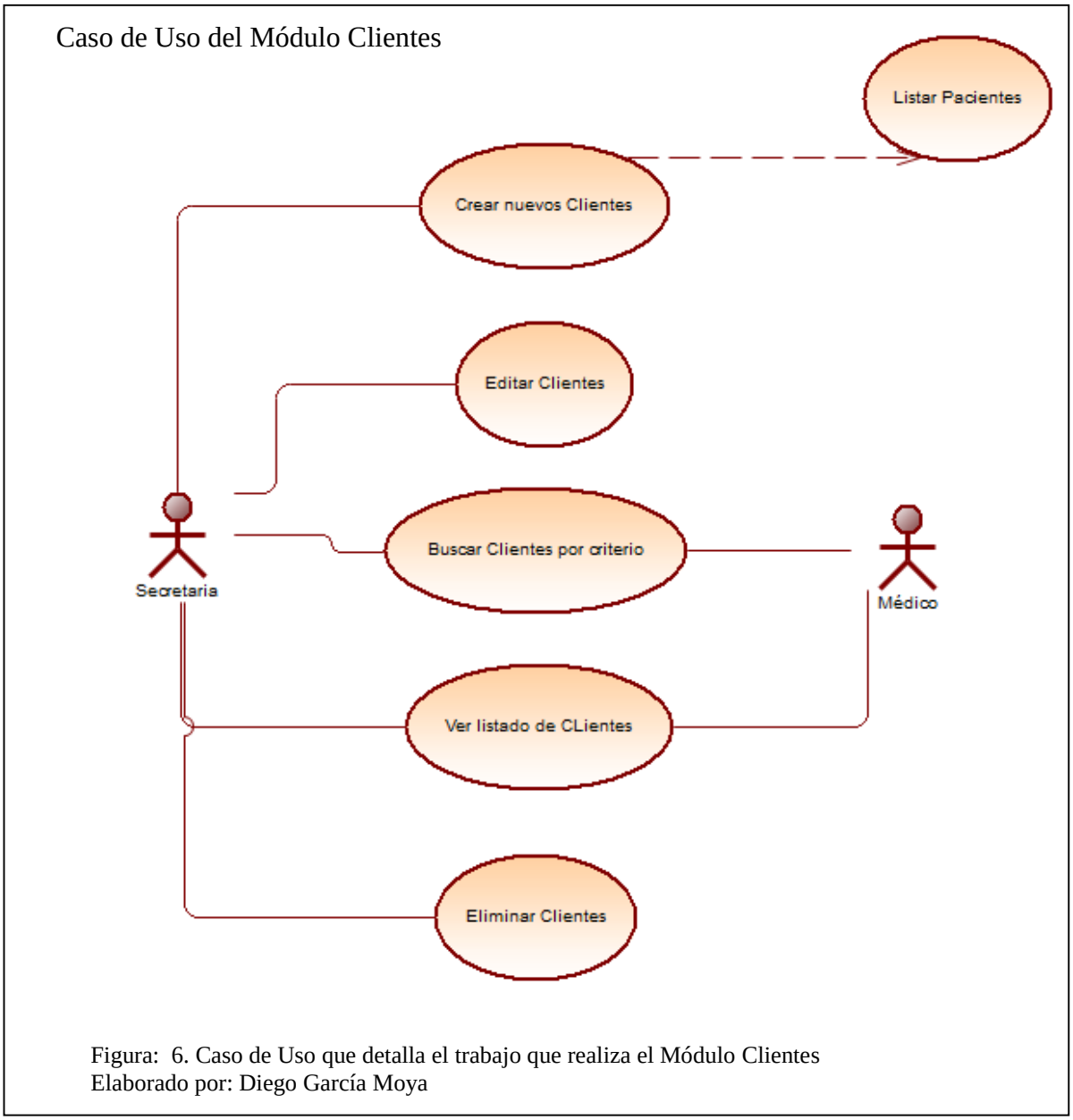


Figura: 6. Caso de Uso que detalla el trabajo que realiza el Módulo Clientes
Elaborado por: Diego García Moya

2.2.1.5 Caso de Uso Funcionalidad módulo Pacientes.

Mediante el diagrama de la figura 7, se representa todas las interacciones de los actores "Secretaria" y "Médico" con los casos de uso para el módulo de Administración de Pacientes, estos actores poseen permisos totales sobre este módulo pudiendo realizar acciones como crear, leer, actualizar, eliminar.

Caso de uso del Módulo Pacientes

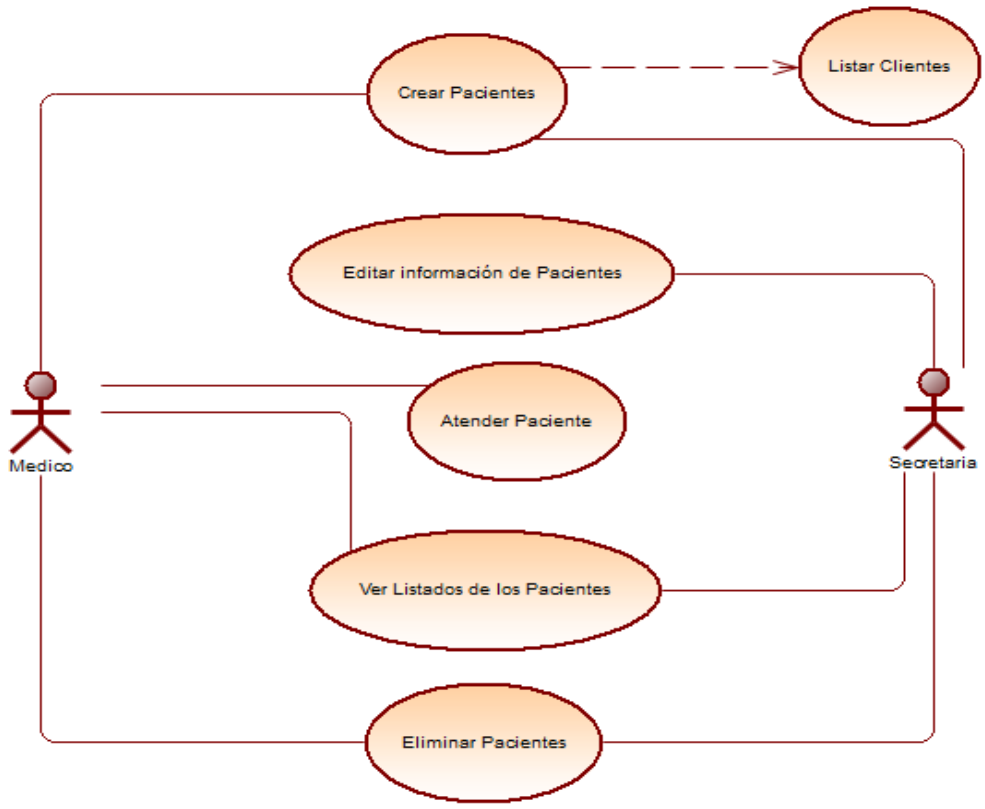


Figura: 7 Caso de Uso que detalla el trabajo que realiza el Módulo Pacientes
Elaborado por: Diego García Moya

2.2.1.6 Caso de Uso Funcionalidad módulo Agendamiento de Citas Médicas.

Mediante el diagrama de la figura 8, se representa todas las interacciones del actor "Secretaria" con los casos de uso para el módulo de Agendamiento de Citas, este actor posee permisos totales sobre este módulo pudiendo realizar acciones como crear, leer, actualizar, eliminar, filtrar las citas, también listar pacientes y visualizar fechas y horas disponibles o no disponibles.

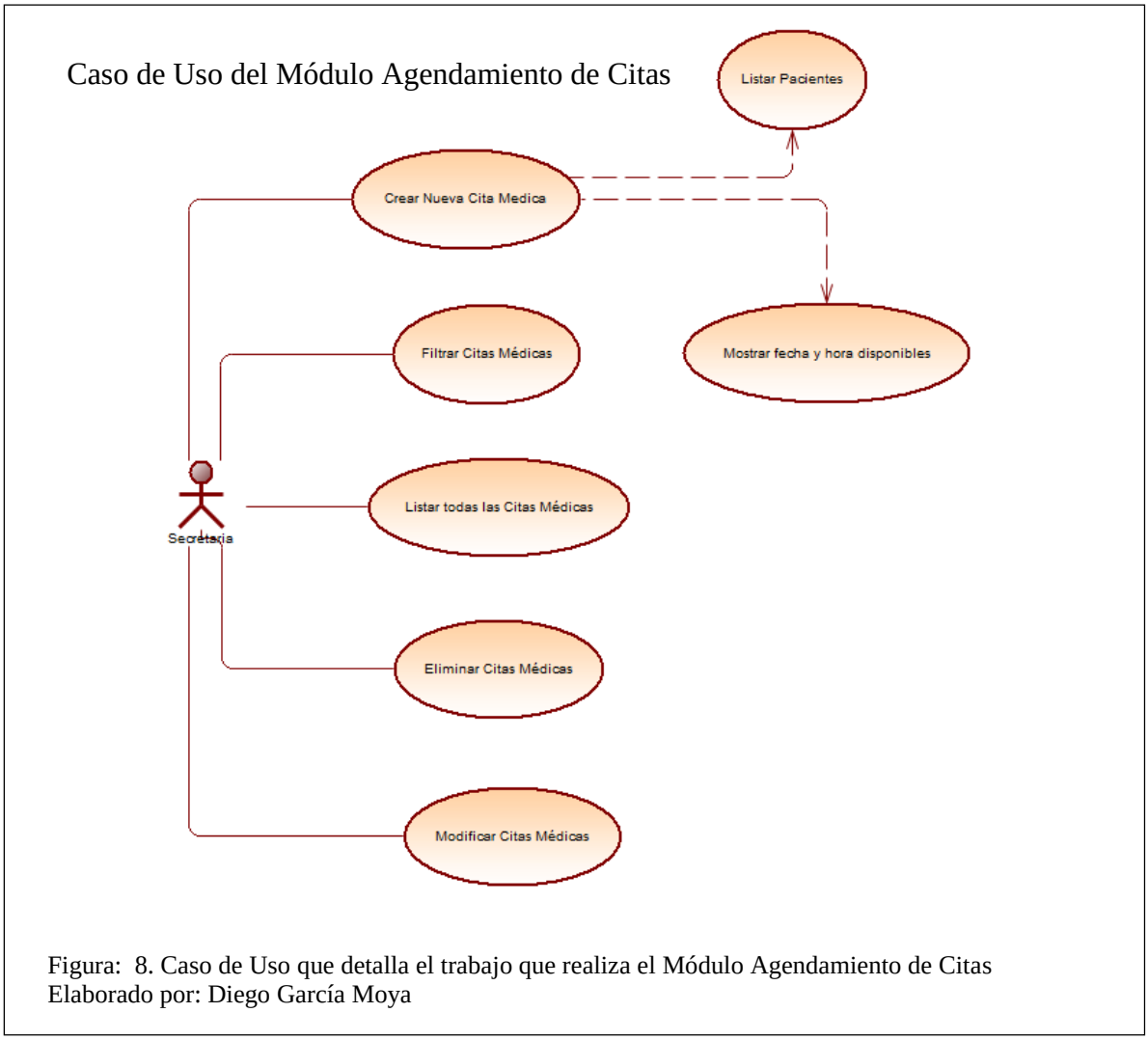


Figura: 8. Caso de Uso que detalla el trabajo que realiza el Módulo Agendamiento de Citas
Elaborado por: Diego García Moya

2.2.1.7 Caso de Uso de la Funcionalidad del módulo Facturación.

Mediante el diagrama de la figura 9, se representa todas las interacciones de los actores "Secretaria", "Médico" y "Vendedor" con los casos de uso para el módulo de Facturación, estos actores poseen permisos específicos sobre este módulo pudiendo realizar acciones como generar Facturas, Procesarlas, imprimirlas o listarlas.

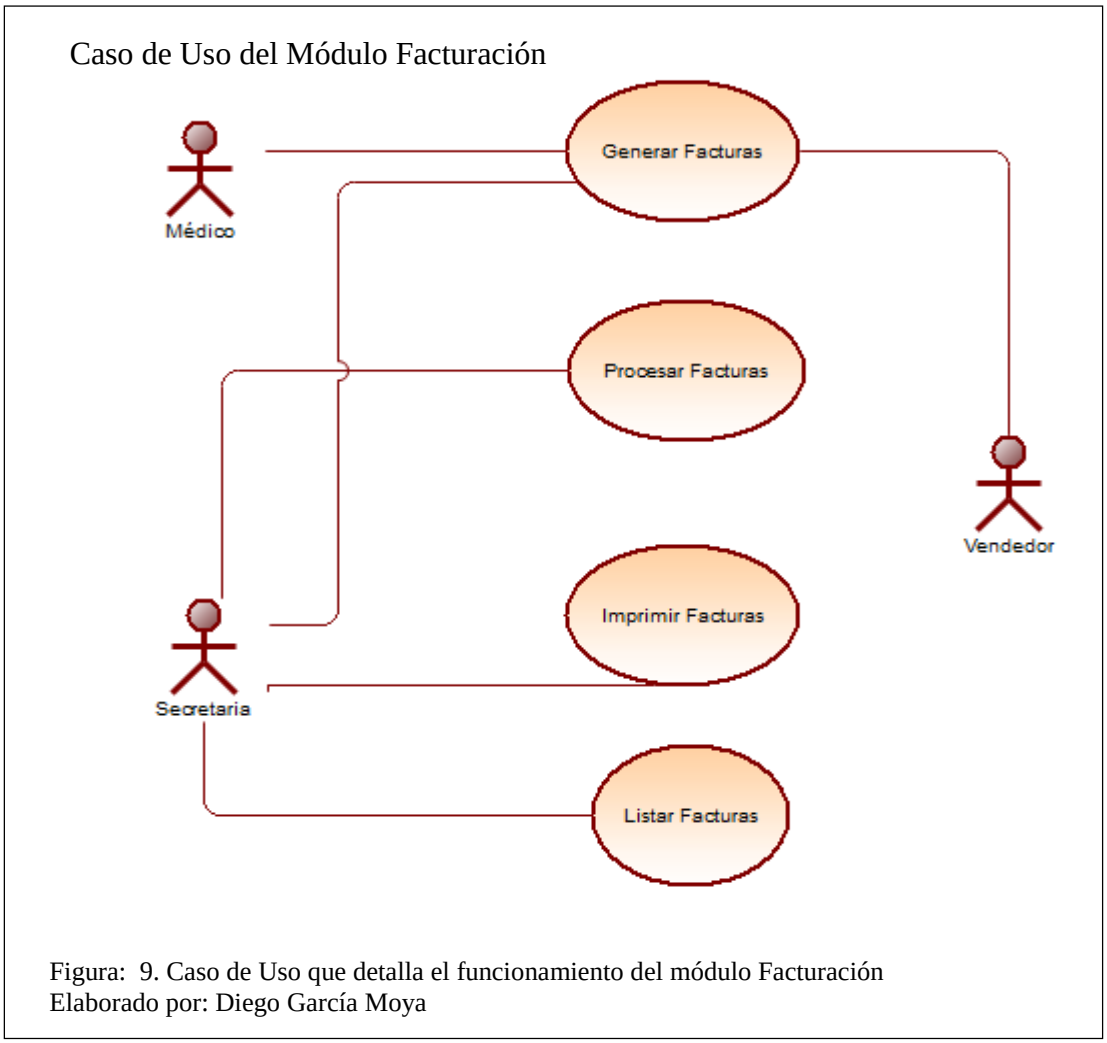


Figura: 9. Caso de Uso que detalla el funcionamiento del módulo Facturación
Elaborado por: Diego García Moya

2.2.1.8 Caso de Uso de la Funcionalidad del módulo Usuarios.

Mediante el diagrama de la figura 10, se representa todas las interacciones del actor "Administrador" con los casos de uso para el módulo de Usuarios, este actor posee permisos totales sobre este módulo pudiendo realizar acciones como crear, leer, actualizar, eliminar, filtrar usuarios.

Caso de Uso del módulo Usuarios

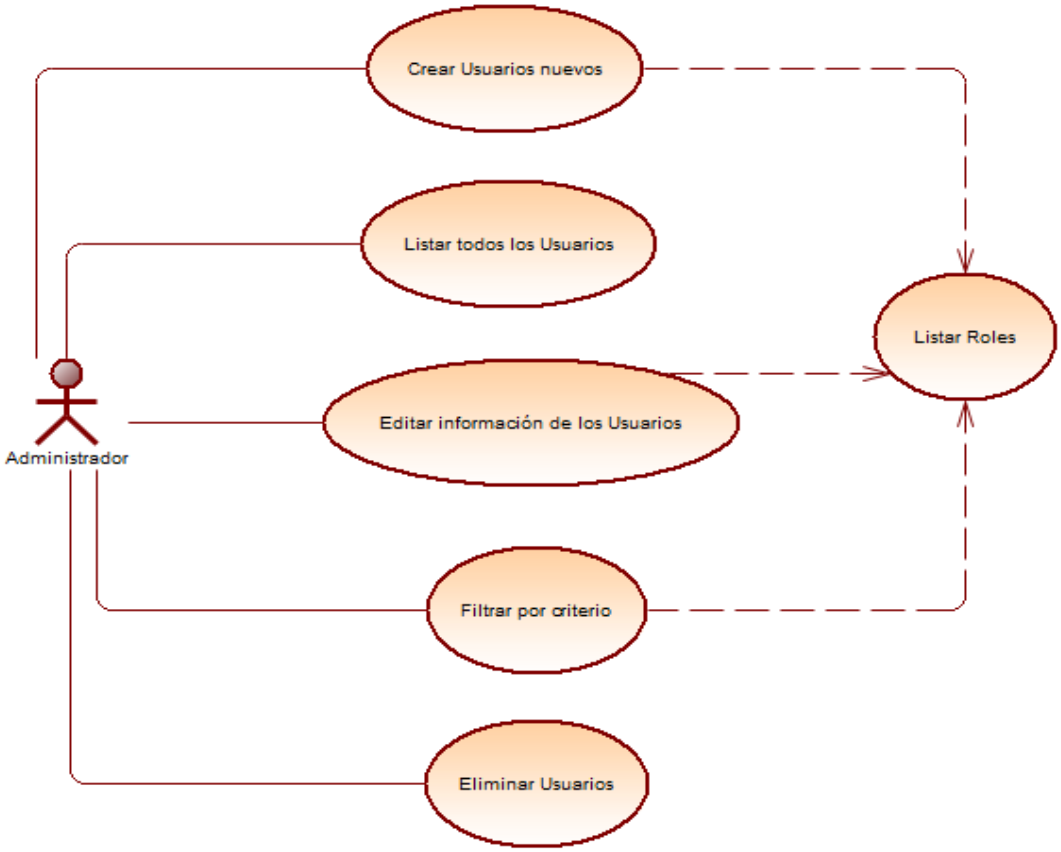


Figura: 10. Caso de Uso que detalla la funcionalidad del módulo Usuarios
Elaborado por: Diego García Moya

2.2.1.9 Caso de Uso de la Funcionalidad del módulo Roles.

Mediante el diagrama de la figura 11, se representa todas las interacciones del actor "Administrador" con los casos de uso para el módulo de Roles, este actor posee permisos totales sobre este módulo pudiendo realizar acciones como crear, leer, actualizar, eliminar, filtrar roles.

Caso de Uso del módulo Roles

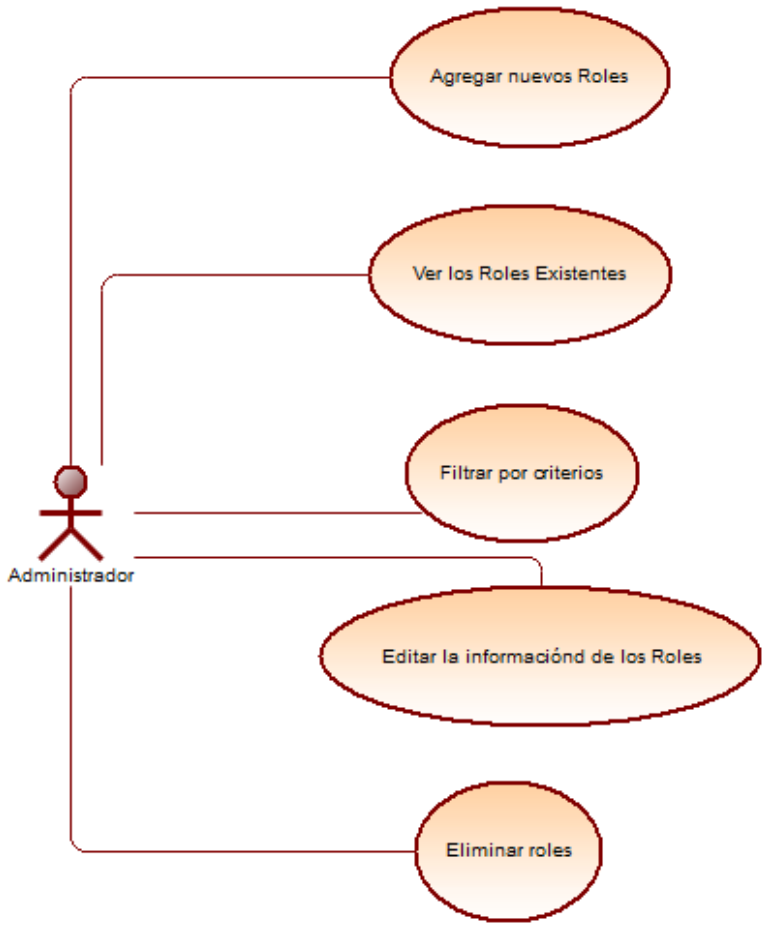


Figura: 11. Caso de uso que detalla la funcionalidad del módulo Roles
Elaborado por: Diego García Moya

2.2.2 Artefactos SCRUM (Product Backlog).

A continuación, se muestran las historias de usuario que se obtuvo en el levantamiento de requerimientos para el sistema mediante varias entrevistas con personal de la veterinaria.

2.2.2.1 Historia de Usuario "Administración de Productos".

La siguiente tabla muestra la historia de usuario Administrar Productos, definiendo los actores, las características y funcionalidades esperadas con relación a la administración de los Productos.

Tabla 32. Historia de Usuario Administración de Productos

Historia de Usuario		
Actor / Rol	**Característica / Funcionalidad**	**Razón / Resultado**
Bodeguero	Registrar productos nuevos cuando los proveedores los entreguen, actualizar el detalle de su información o stock existente, además se requiere la capacidad de eliminarlos si es el caso, buscarlos mediante nombre del producto o por categoría, o simplemente listar todos los productos existentes. recibir emails que notifiquen en el momento en que uno o	Para mantener actualizado el stock y poder tomar decisiones de adquirir o no productos, y lo más importante para poder despachar ventas verificando si las existencias de determinado producto satisfacen un pedido, Buscarlos para ver detalles de la información del producto

	varios productos estén próximos a agotarse.	
Criterios de Aceptación		
Contextos	1. En el caso en que el bodeguero no ingrese los datos completos al intentar registrar un producto nuevo. 2. En el caso en que el encargado filtre un producto usando un criterio no existente. 3. En el caso en que uno o varios productos se aproximen a su valor mínimo de existencias en bodega, ya sea por ventas u otros motivos.	
Eventos	1. Cuando él quiere guardar el producto nuevo. 2. Cuando el intente visualizar un producto por criterio. 3. Cuando se actualice el valor mínimo de existencias por ventas o por otro motivo	
Resultados	1. El sistema le presenta una alerta indicando los campos que son obligatorios. 2. El sistema le responderá con un mensaje indicando lo que ha ocurrido. 3. El sistema enviara de manera automática un email notificando lo que ocurre con el o los productos.	

Nota: Esta tabla contiene la historia de Usuario: Administración de Productos

2.2.2.2 Historia de Usuario "Administración de Clientes".

EN la tabla siguiente se describen las funciones y características que se debe realizar con respecto a la administración de clientes, se definen los actores, y también los criterios de aceptación y respuestas esperadas.

Tabla 33. Historia de Usuario Administración de Clientes

Historia de Usuario		
Actor / Rol	**Característica / Funcionalidad**	**Razón / Resultado**
Secretaria, Médico	Registrar la información completa de nuevos clientes, actualizar información si fuera el caso, o eliminarlos, además se requiere buscarlos por apellido o número de identificación, también aquí se deseo poder indicar que paciente o pacientes pertenecen a un cliente.	Registrar clientes nuevos para llevar de manera ordenada su información, buscar por criterio para encontrar un cliente de manera rápida, Editar información para cambiar en los casos que la información del cliente cambie y así mantener una base actualizada, asociar paciente cliente para identificar de manera adecuada los tratamientos o atenciones de cada uno
Criterios de Aceptación		
Contextos	1. En el caso en que el usuario ingrese información incompleta. 2. En el caso en que el usuario realice una búsqueda cuyo criterio no exista.	
Eventos	1. Cuando el intente guardar el cliente nuevo. 2. Cuando intente filtrar un cliente por criterio	

Resultados	1. El sistema le notificara mediante una alerta indicando lo que ha ocurrido. 2. El sistema le mostrara un mensaje indicando lo ocurrido

Nota: Esta tabla contiene la historia de Usuario: Administración de Clientes

2.2.2.3 Historia de Usuario "Administración de Pacientes".

A continuación, en la tabla se detallan los usuarios que requieren ejecutar distintas acciones en torno a la administración de los pacientes.

Tabla 34. Historia de Usuario Administración de Pacientes

Historia de Usuario		
Actor / Rol	**Característica / Funcionalidad**	**Razón / Resultado**
Médico, Secretaria	Registrar la información completa de pacientes nuevos, indicar cuál es su propietario o cliente, también se desea poder modificar o eliminar la información del paciente, Además se requiere que el sistema automáticamente calcule la edad animal del paciente para los casos de canes o felinos.	Manipular la información para mantener un adecuado registro de todos los pacientes y poder acceder a esa información de manera oportuna. Calcular la edad para conocer la edad canina o felina con lo que permitirá al médico realizar un análisis y valoración correctos.
Criterios de Aceptación		
Contextos	1. En el caso en que se ingresen información incorrecta.	

	2. En el caso en que el usuario no complete la información requerida en el formulario.
Eventos	1. Cuando el usuario ingrese o edite información del paciente. 2. Cuando el usuario requiera crear nuevos clientes.
Resultados	1. El sistema presentara un mansaje informando lo ocurrido. 2. El sistema le presentara una alerta sobre lo ocurrido.

Nota: Esta tabla contiene la historia de Usuario: Administración de Pacientes

2.2.2.4 Historia de Usuario "Atención Médica".

Los médicos son quienes requieren la capacidad de registrar datos e información relacionada con una atención médica.

Tabla 35. Historia de Usuario Atención Médica

Historia de Usuario		
Actor / Rol	**Característica / Funcionalidad**	**Razón / Resultado**
Médico	Atender a los pacientes que asisten a consulta pudiendo en la misma realizar la toma de constantes Fisiológicas, compararlas con las de la última visita, ver históricos de atenciones médicas, vacunación y desparasitación, exámenes realizados, atenciones en peluquería. Se requiere poder crear un	Registrar información de la atención medica al paciente para, alimentar la historia clínica del paciente y así poder diagnosticarlo efectivamente mediante la visualización de los históricos. Crear e imprimir tratamientos para entregarlo al cliente y siga las instrucciones correctamente.

	tratamiento para el paciente e imprimirlo, también registrar un control de vacunación - desparasitación	Registrar Vacunaciones o desparasitaciones para llevar un adecuado control de cada paciente.
Criterios de Aceptación		
Contextos	1. En el caso en que el médico ingrese un ítem erróneo para el tratamiento. 2. En el caso que el médico quiera agregar ítems tanto al tratamiento o al control de vacunación/desparasitación que ya hayan sido creados previamente en esa consulta. 3. En el caso en que el médico ingrese valores incorrectos en los diferentes campos del formulario.	
Eventos	1. Cuando él se encuentre creando un tratamiento. 2. Cuando él necesite añadir un ítem que olvido en el tratamiento o control de vacunación/desparasitación. 3. Cuando registre información referente a la consulta actual.	
Resultados	1. El sistema le permitirá eliminar o editar el ítem. 2. El sistema mostrara los ítems actuales y permitirá que se agreguen más. 3. El sistema le presentara una alerta informándole sobre lo que está ocurriendo.	

Nota: Esta tabla contiene la historia de Usuario: Atención Médica

2.2.2.5 Historia de Usuario "Facturación".

Un requerimiento de la veterinaria es la funcionalidad de generar y emitir facturas cuando sea necesario, en la siguiente tabla se describen los actores y características del

funcionamiento del sistema frente a distintos eventos donde el sistema debe responder de acuerdo a los criterios que se describen en la tabla 36.

Tabla 36. Historia de Usuario Facturación

Historia de Usuario		
Actor / Rol	**Característica / Funcionalidad**	**Razón / Resultado**
Vendedor, Secretaria, Médico	Generar facturas para cuando se vendan productos o se realicen atenciones médicas, imprimirlas para entregarlas a los clientes.	La facturación permitirá mejorar el control financiero, también permitirá llevar un control de las existencias que se actualizarán al momento de realizar ventas o atenciones medicas
Criterios de Aceptación		
Contextos	1. En el caso cuando un usuario quiera reimprimir una factura. 2. En el caso que el vendedor necesite añadir nuevos ítems o eliminarlos.	
Eventos	1. Cuando el cliente solicite una reimpresión de la factura. 2. Cuando un cliente solicite agregar un ítem extra.	
Resultados	1. El sistema permitirá buscar el cliente por número de cedula y mostrará todas sus facturas para poder imprimirlas. 2. EL sistema permitirá agregar o eliminar ítems de facturas que hayan sido creadas pero no procesadas (cobradas)	

Nota: Esta tabla contiene la historia de Usuario: Facturación

2.2.2.6 Historia de Usuario "Administración de Usuarios".

A continuación, se muestra en detalle las características funcionales que el usuario requiere al momento de administrar usuarios del sistema.

Tabla 37. Historia de Usuario Administración de Usuarios

Historia de Usuario		
Actor / Rol	**Característica / Funcionalidad**	**Razón / Resultado**
Administrador	Gestionar los usuarios del sistema, poder agregar datos sobre cada usuario, asignarles credenciales para el acceso al sistema y darles permisos para determinadas partes del sistema determinadas por el rol que desempeñen en la veterinaria	Para mejorar la seguridad y el desempeño de las actividades de cada usuario con la concesión de roles y permisos sobre el sistema, lo cual les permitirá centrarse en su rol.
Criterios de Aceptación		
Contextos	1. En el caso en que un usuario requiera de permisos totales sobre el sistema. 2. En el caso en que un usuario se encuentre de vacaciones o este fuera de la institución por un periodo determinado de tiempo.	
Eventos	1. Cuando un administrador requiera crear un usuario con acceso total a todos los módulos del sistema.	

	2. Cuando un administrador quiera suspender temporalmente a un usuario.
Resultados	1. El sistema permitirá la creación de un usuario con permisos especiales “all-acces”. 2. El sistema permitirá que un administrador ponga en estado suspendido a un usuario lo cual le quita todo permiso sobre el sistema.

Nota: Esta tabla contiene la historia de Usuario: Administración de Usuarios.

2.2.2.7 Historia de Usuario “Agendar Citas Médicas”.

Al momento de agendar una cita médica se requiere que el sistema tenga la capacidad de verificar la disponibilidad de los médicos en determinado horario y fecha para adherir a su agenda una cita médica nueva, el sistema debe responder a varios eventos con el mensaje adecuado dependiendo del criterio de aceptación especificado en la historia de usuario.

Tabla 38. Historia de Usuario Agendar Citas Médicas

Historia de Usuario		
Actor / Rol	**Característica / Funcionalidad**	**Razón / Resultado**
Secretaria, Médico	Agendar citas médicas para los pacientes, de manera tal que no se crucen horarios ni fechas de agendas similares. También se deben enviar recordatorios vía email de estas	Agendamiento de citas médicas para evitar que se crucen fechas y horas que ya hayan sido agendadas. Envío de recordatorios para que los usuarios no olviden la cita que

	citas médicas un día antes de que la hora y fecha agendada	tienen agenda, y de esta manera la atención fluya correctamente.
Criterios de Aceptación		
Contextos	1. En el caso de que el usuario quiera agendar en una fecha u hora que ya no están disponibles.	
Eventos	1. Agendamiento de citas medicas	
Resultados	1. El sistema mostrara un mensaje indicando que la fecha u hora ya no están disponibles.	

Nota: Esta tabla contiene la historia de Usuario: Agendar Citas Médicas.

2.3 Bases de Datos

2.3.1 Esquema Físico de la Base de Datos.

La figura muestra el esquema completo del modelo Físico de la Base de Datos.

Diagrama Físico de la Base de Datos

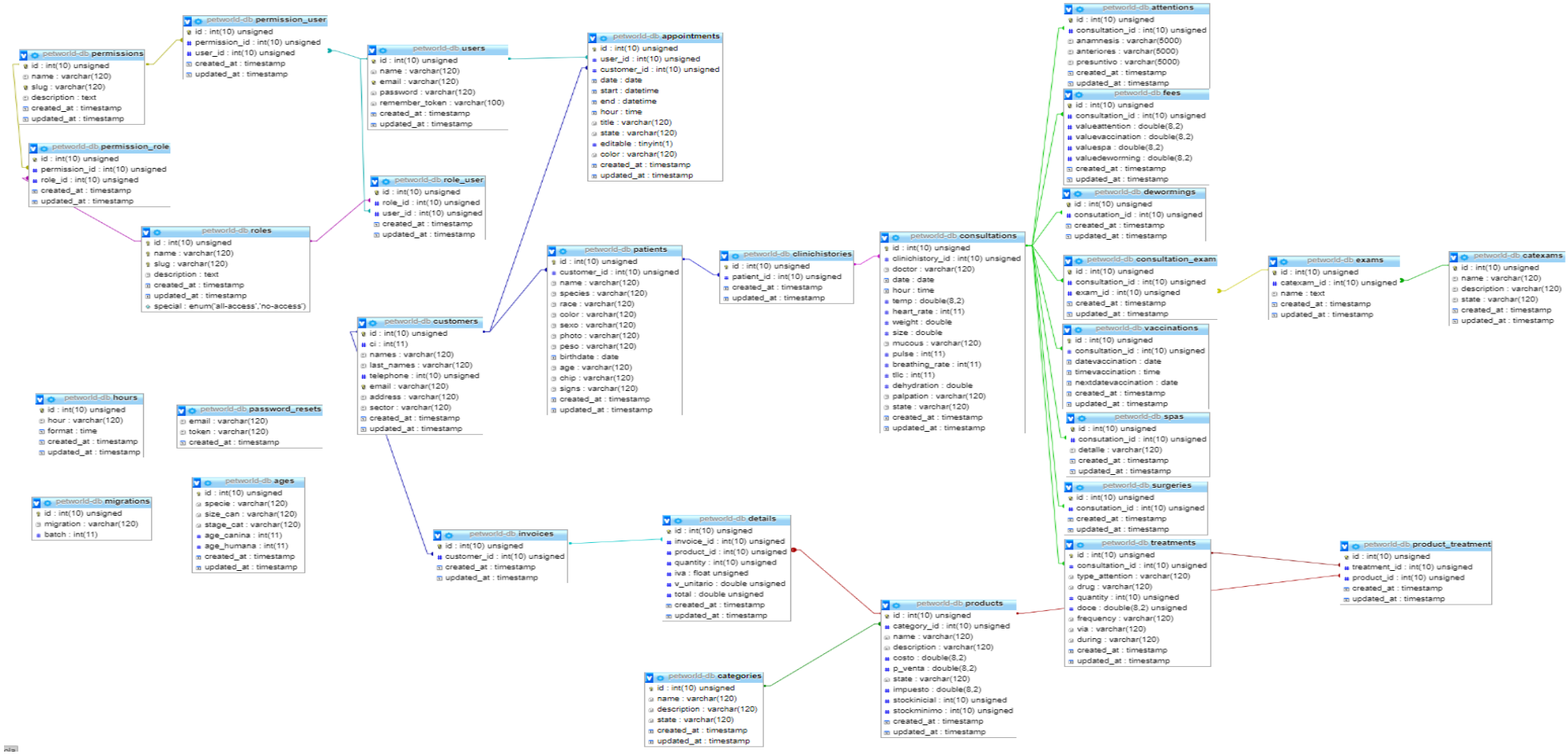


Figura: 12. Diagrama Físico de la Base de Datos
Elaborado por: Diego García Moya

2.4 Interfaces Graficas de Usuario

2.4.1 Prototipos del Módulo Productos.

En la figura 13, se muestra el diseño realizado como prototipo para la interfaz del usuario para listar los productos de bodega, se muestra una tabla con información relevante de los productos y un botón de acciones básicas sobre los registros.

Listado de Productos

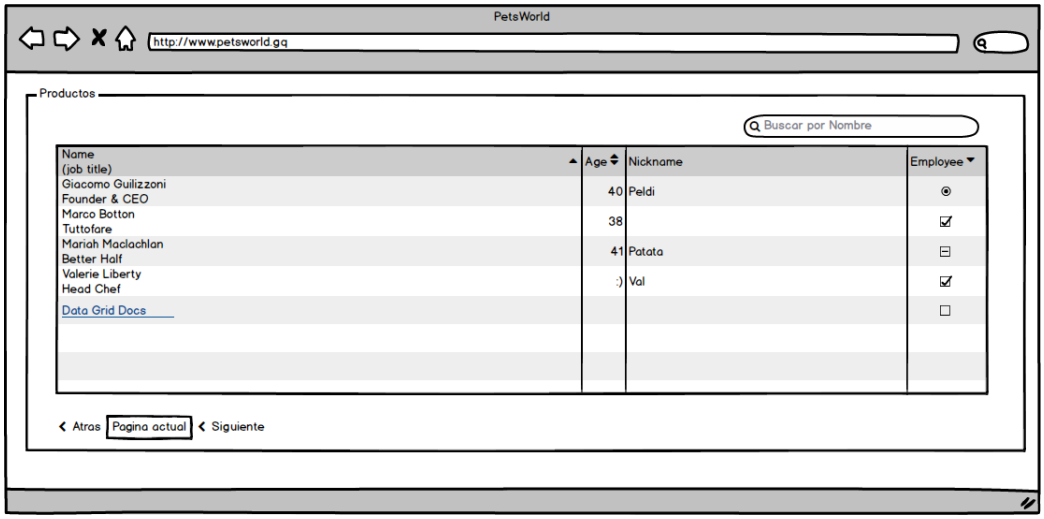


Figura: 13. Prototipo de la vista Productos - Listado del Productos
Elaborado por: Diego García Moya

2.4.2 Prototipos del Módulo Clientes.

Para la interfaz de usuario en la que se posibilita listar los clientes registrados en el sistema se ha diseñado el prototipo que se muestra en la figura 14, se incluye una tabla en la que se mostrara información importante del cliente, además un botón con acciones básicas sobre los registros.

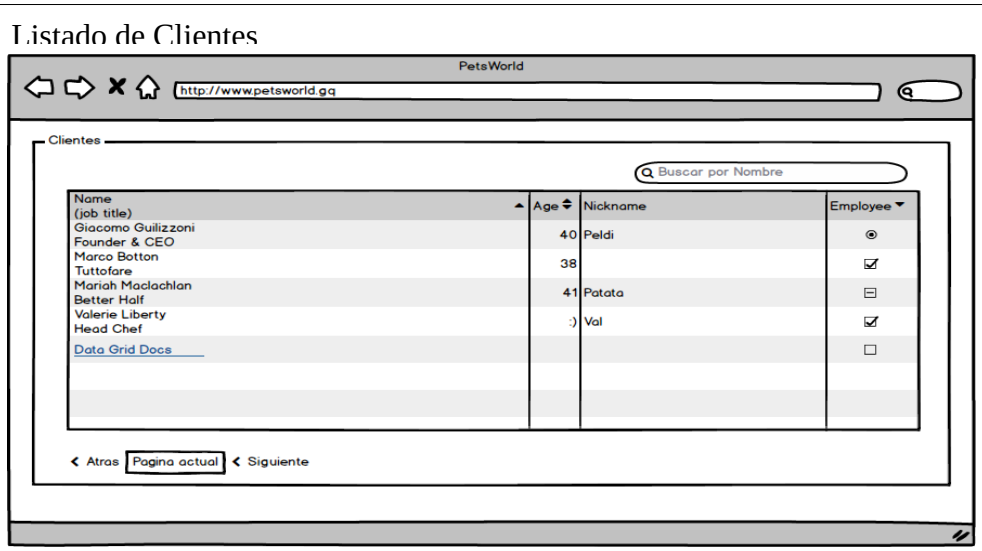


Figura: 14. Prototipo de la vista Clientes - Listado de Clientes
Elaborado por: Diego García Moya

2.4.3 Prototipos del Módulo Pacientes.

En la figura 15, se muestra el prototipo realizado para la interfaz en la que el usuario podrá visualizar una tabla con la información relevante de los pacientes, editar, crear y eliminar.

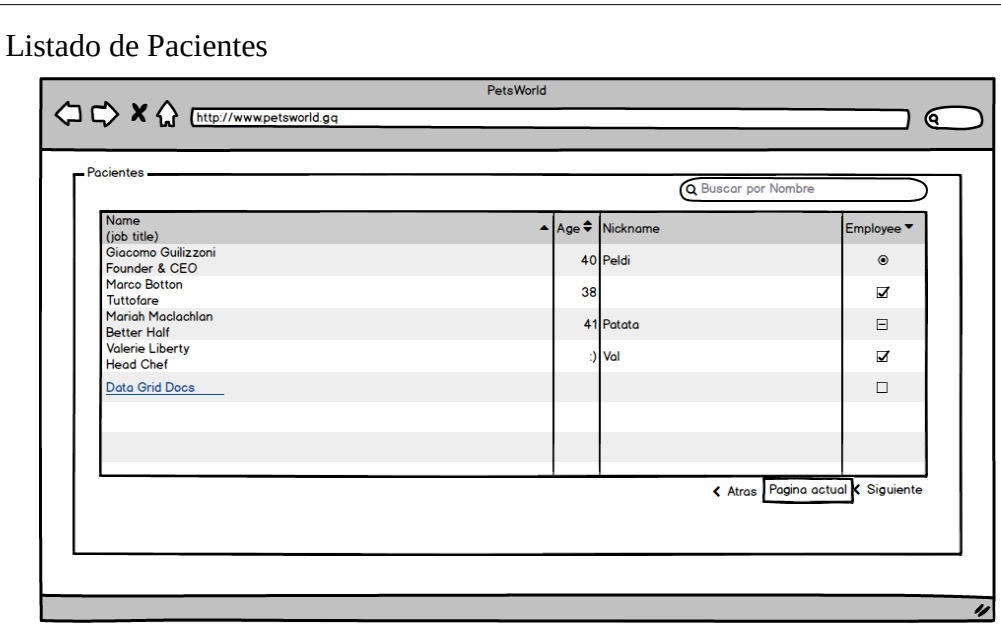


Figura: 15. Prototipo de la vista Pacientes - Listado de Pacientes
Elaborado por Diego García Moya

CAPÍTULO 3

IMPLEMENTACIÓN

El software se ha creado utilizando el Framework Laravel el mismo que como lenguaje base tiene a PHP, este framework instalar un proyecto crea el directorio de archivos implementando la arquitectura MVC (Modelo Vista Controlador), de tal manera que la codificación para las interfaces graficas de usuario permanezca independiente, separada de la codificación para el backend, y de la misma manera la codificación que interactúa con la base de datos. Esta arquitectura hace que el sistema sea escalable y de fácil mantenimiento. Al estar el sistema separado en componentes es necesario un medio de comunicación entre ellos, para lo cual se utilizan las tecnologías Axios y Eloquent, que se mencionan en el marco teórico. Este medio de comunicación tendrá dos canales uno de la vista al controlador y otro del controlador al modelo, a continuación, se describe el trabajo que realiza este medio de comunicación.

Para la comunicación entre la vista y el controlador se hace uso de la tecnología AJAX mediante Axios de Laravel para gestionar las peticiones XMLHttpRequest desde la vista hacia el controlador y viceversa.

La comunicación entre el Controlador y el Modelo se lleva a cabo mediante el ORM de Laravel “Eloquent”, que gracias a su capacidad de realizar un mapeo objeto-relacional se logra tener acceso a la base de datos con todas las prestaciones que ofrece un lenguaje Orientado a Objetos.

Para el envío de recordatorio vía email a los clientes se configuro “Cron” que es un demonio que se encuentra presente en el Sistema Operativo del Servidor, que para este caso es un Linux Ubuntu Server 18.04, en la configuración de “Cron” se debe

especificar la ruta del “Command” que se ejecutará de acuerdo a la periodicidad que se indica en el mismo archivo.

Todas las Vistas están implementadas con la tecnología Vue JS que se describe en el marco teórico, con el objetivo de dotarlas de dinamismo durante la interacción del usuario.

Para el flujo de esta información primeramente se debe especificar las url’s que se manejaran en determinada acción, por ejemplo, para el registro de mascotas se crea la url “patients/store” en el archivo “web” de las rutas en Laravel, la ruta creada consta de dos partes donde la primera hace referencia al Controlador a llamarse, en este caso es es “PatientsController” y el método llamado es “store”. Posteriormente se crea la Interfaz de usuario para el ingreso de mascotas usando HTML para la maquetación de la página, Bootstrap para los diseños CSS, y VueJs para la comunicación de datos mediante “Axios”, donde mediante el verbo Http “POST” se envía serializada la data a ser almacenada, esta data viaja dentro de la petición y es capturada por el método “store” en el Controlador, el Controlador valida que la data recibida sea la esperada de acuerdo a reglas de validación especificadas en el método, si todo es correcto el controlador utiliza “Eloquen” para crear una consulta a la base de datos enviándole en la sentencia los datos a almacenar, al realizar el grabado del registro se pueden esperar respuestas positivas o negativas, esta respuesta es enviada mediante “Response” al método de “Axios” aquí se evalúa la respuesta y se entrega al usuario mensajes de información, alerta o precaución describiendo lo ocurrido en el proceso de grabado en la base de datos, esto último se realiza con la utilización de “Toastr” que es la librería de JavaSctript que gestiona mensajes en el Frontend.

Lo descrito anteriormente es el funcionamiento y flujo básico de la información en el presente sistema tomando en cuenta que se implementas las herramientas descritas a continuación.

3.1 Implementación del Entorno de Desarrollo

El sistema está pensado para funcionar en un entorno Web y especialmente en un ecosistema económico, por tal motivo se requiere de la preparación de varios elementos para lograr el objetivo, se requiere de un servidor web, para alojar los archivos del sistema y la base de datos que usara el mismo. A continuación, se detalla cada elemento del entorno.

3.1.1 Servidor Web.

El servidor web es una virtualización de un equipo que posee las siguientes características:

- Memoria Ram 1GB
- Una vCPU
- Disco Duro 25 GB
- Sistema Operativo Linux Ubuntu 18.04

Mediante la herramienta LAMP se instaló el siguiente software:

- Servidor Web Apache 2.4.29
- Motor de Base de Dartos MySQL 5.7.25
- PhpMyAdmin para la administración de la Base de Datos
- El lenguaje de programación PHP en la versión 7.2.17

3.1.2 Herramientas y Dependencias.

A continuación, se listan las herramientas, frameworks y librerías instalados para crear el entorno de desarrollo deseado para el sistema.

- Node JS 11.4.0
- Composer 1.7.2
- Laravel 5.6.33
- VueJS 2.5.7
- Bootstrap 4.0
- JQuery 3.2
- OpenSSL PHP Extension
- PDO PHP Extension
- Mbstring PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension
- Ctype PHP Extension
- JSON PHP Extension

3.1.3 Diagrama de Clases.

Diagrama de Clases General

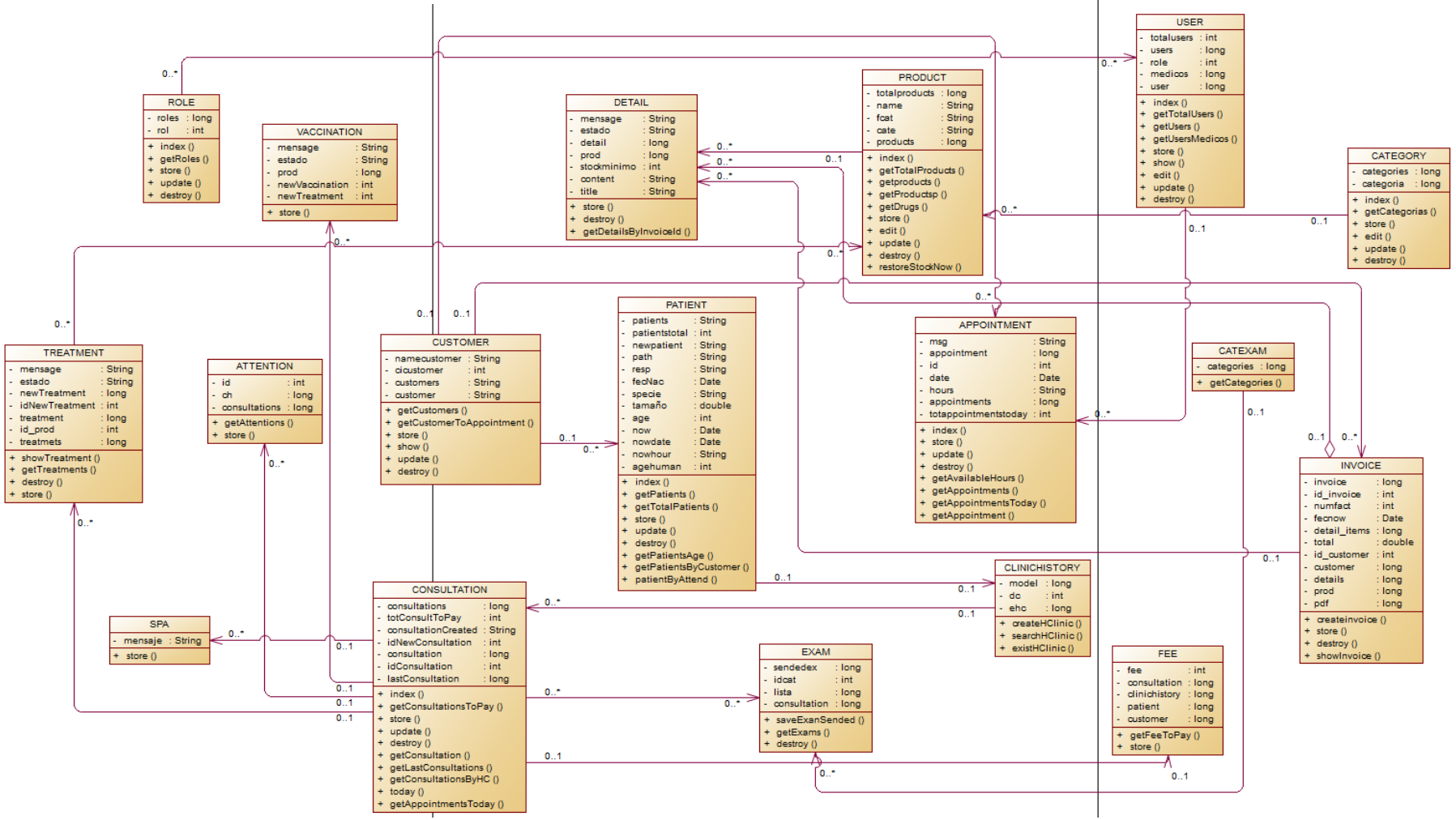


Figura: 16. Diagrama de Clases del Sistema
Elaborado por: Diego García Moya

3.1.4 Diccionario de Datos.

La tabla No.39 se detalla todas las entidades que conforman la Base de Datos para el sistema, en ella se indica el nombre de cada tabla, sus campos y el tipo de dato.

Tabla 39. Diccionario de Datos

Nombre de Tabla	Campo	Tipo
ages	id	int(10)
	specie	varchar(120)
	size_can	varchar(120)
	stage_cat	varchar(120)
	age_canina	int(11)
	age_humana	int(11)
	created_at	Timestamp
	updated_at	Timestamp
appointments	id	int(10)
	user_id	int(10)
	customer_id	int(10)
	date	Date
	start	Datetime
	end	Datetime
	hour	Time
	title	varchar(120)
	state	varchar(120)
	editable	tinyint(1)
	color	varchar(120)
	created_at	Timestamp
	updated_at	Timestamp
attentions	id	int(10)
	consultation_id	int(10)
	anamnesis	varchar(500)
	anteriores	varchar(500)
	presuntivo	varchar(500)
	created_at	Timestamp
	updated_at	Timestamp
categories	id	int(10)
	name	varchar(120)
	description	varchar(120)
	state	varchar(120)
	created_at	Timestamp
	updated_at	Timestamp
catexams	id	int(10)

	name	varchar(120)
	description	varchar(120)
	state	varchar(120)
	created_at	Timestamp
	updated_at	Timestamp
clinichistories	id	int(10)
	patient_id	int(10)
	created_at	Timestamp
	updated_at	Timestamp
consultation_exam	id	int(10)
	consultation_id	int(10)
	exam_id	int(10)
	created_at	Timestamp
	updated_at	Timestamp
consultations	id	int(10)
	clinichistory_id	int(10)
	doctor	varchar(120)
	date	Date
	hour	Time
	temp	double(8.2)
	heart_rate	int(11)
	weight	Double
	size	Double
	mucous	varchar(120)
	pulse	int(11)
	breathing_rate	int(11)
	tilc	int(11)
	dehydration	Double
	state	varchar(120)
	created_at	Timestamp
	updated_at	Timestamp
customers	id	int(10)
	ci	int(11)
	names	varchar(120)
	last_names	varchar(120)
	telephone	int(10)
	email	varchar(120)
	address	varchar(120)
	sector	varchar(120)
	created_at	Timestamp
	updated_at	Timestamp
details	id	int(10)
	invoice_id	int(10)

	product_id	int(10)
	quantity	int(10)
	iva	float
	v_unitario	double
	total	double
	created_at	timestamp
	updated_at	timestamp
dewormings	id	int(10)
	consutation_id	int(10)
	created_at	timestamp
	updated_at	timestamp
exams	id	int(10)
	catexam_id	int(10)
	name	text
	created_at	timestamp
	updated_at	timestamp
fees	id	int(10)
	consultation_id	int(10)
	valueattention	double(8,2)
	valuevaccination	double(8,2)
	valuespa	double(8,2)
	valuedeworming	double(8,2)
	created_at	timestamp
	updated_at	timestamp
hours	id	int(10)
	hour	varchar(120)
	format	time
	created_at	timestamp
	updated_at	timestamp
invoices	id	int(10)
	consultation_id	int(10)
	created_at	timestamp
	updated_at	timestamp
migrations	id	int(10)
	migration	varchar(120)
	batch	int(11)
password_resets	email	varchar(120)
	token	varchar(120)
	created_at	timestamp
patients	id	int(10)
	customer_id	int(10)
	name	varchar(120)
	species	varchar(120)

	race	varchar(120)
	color	varchar(120)
	sexo	varchar(120)
	photo	varchar(120)
	peso	varchar(120)
	birthdate	date
	age	varchar(120)
	chip	varchar(120)
	sings	varchar(120)
	created_at	timestamp
	updated_at	timestamp
permission_role	id	int(10)
	permission_id	int(10)
	role_id	int(10)
	created_at	timestamp
	updated_at	timestamp
permission_user	id	int(10)
	permission_id	int(10)
	user_id	int(10)
	created_at	timestamp
	updated_at	timestamp
permissions	id	int(10)
	name	varchar(120)
	slug	varchar(120)
	description	tex
	created_at	timestamp
	updated_at	timestamp
product_treatment	id	int(10)
	treatment_id	int(10)
	product_id	int(10)
	created_at	timestamp
	updated_at	timestamp
products	id	int(10)
	category_id	int(10)
	name	varchar(120)
	description	varchar(120)
	costo	double(8,2)
	p_venta	double(8,2)
	state	varchar(120)
	impuesto	double(8,2)
	stockinicial	int(10)
	stockminimo	int(10)
	created_at	timestamp

	updated_at	timestamp
role_user	id	int(10)
	role_id	int(10)
	user_id	int(10)
	created_at	timestamp
	updated_at	timestamp
roles	id	int(10)
	name	varchar(120)
	slug	varchar(120)
	description	tex
	created_at	timestamp
	updated_at	timestamp
	special	enum('all-access', 'no-access')
spas	id	int(10)
	consulation_id	int(10)
	detalle	varchar(120)
	created_at	timestamp
	updated_at	timestamp
surgeries	id	int(10)
	consulation_id	int(10)
	created_at	timestamp
	updated_at	timestamp
treatments	id	int(10)
	consulation_id	int(10)
	type_attention	varchar(120)
	drug	varchar(120)
	quantity	int(10)
	doce	double(8,2)
	frequency	varchar(120)
	via	varchar(120)
	during	varchar(120)
	created_at	timestamp
	updated_at	timestamp
users	id	int(10)
	name	varchar(120)
	email	varchar(120)
	password	varchar(120)
	remember_token	varchar(100)
	created_at	timestamp
	updated_at	timestamp
vaccinations	id	int(10)

	consltation_id	int(10)
	datevaccination	date
	timevaccination	time
	nextdatevaccination	date
	created_at	timestamp
	updated_at	timestamp

Nota: En la Tabla 39 se detallan las entidades y atributos de la Base de Datos.

PRUEBAS DEL SISTEMA

4.1 Pruebas del Sistema

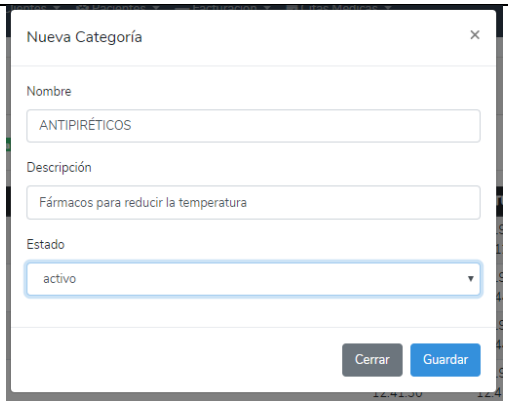
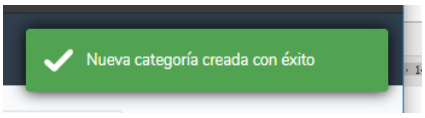
Para la realización de las pruebas del sistema se han realizado tres tipos de pruebas:

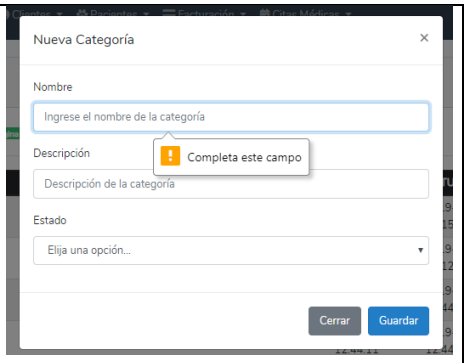
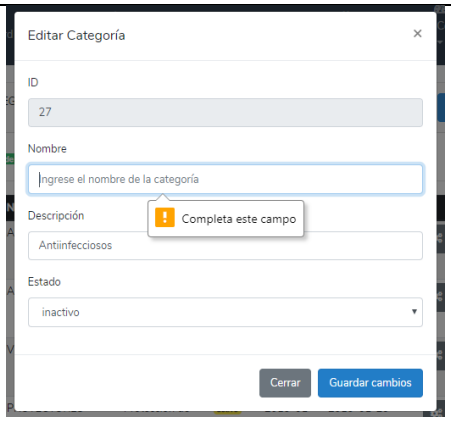
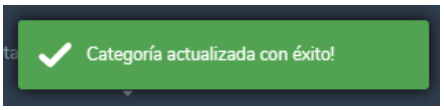
- Pruebas de Caja Negra.
- Pruebas de Rendimiento
- Pruebas de Carga
- Pruebas de Estrés

A continuación, se presenta en la tabla 40, una descripción de la prueba mostrando la funcionalidad ejecutada el tipo de entrada que el usuario entrega al sistema y la salida en respuesta a dicha entrada.

4.1.1 Pruebas de Caja Negra.

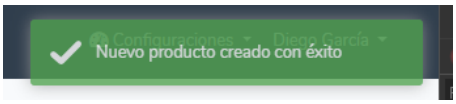
Tabla 40. Pruebas de Caja Negra del módulo de Categorías

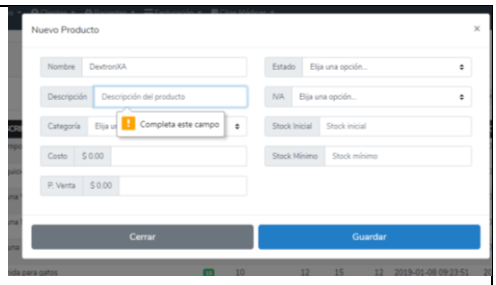
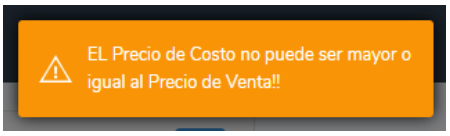
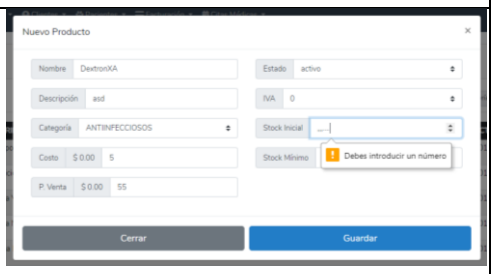
Func	**Tipo de entrada**	**Salida**
Crear Categoría	 Entrada válida	 Mensaje de información.

Crear Categoría	Entrada incompleta	 Mensaje de información.
Editar Categoría	Entrada incompleta	 Mensaje de Información
Editar Categoría	Entrada Valida	 Mensaje de información

Nota: Esta tabla contiene las pruebas de caja negra realizadas al módulo de categorías.

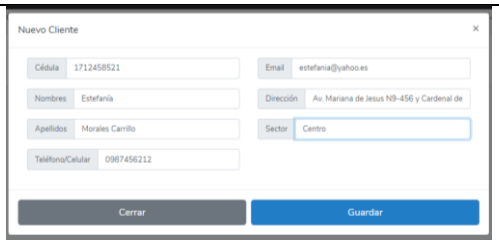
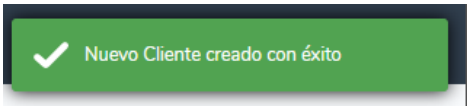
Tabla 41. Pruebas de Caja Negra del módulo Productos

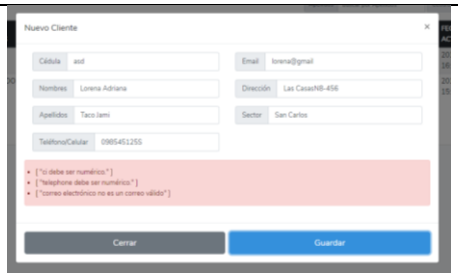
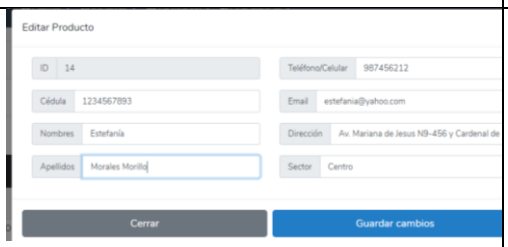
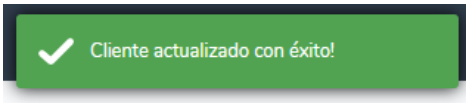
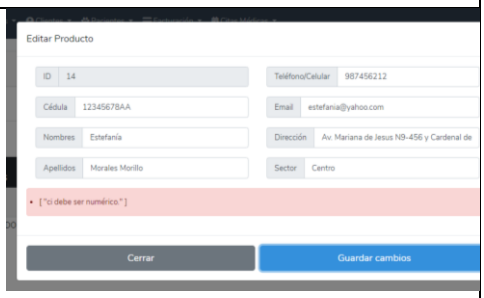
Func	**Tipo de entrada**	**Salida**
Crear Producto	Entrada valida	 Mensaje de información

Crear Producto	Entrada incompleta	 Mensajes de información
Crear/Editar Producto	 Entrada no valida, valor de costo mayo que el precio de venta	 Mensaje de Advertencia
Crear/ Editar Producto	Entrada no valida	 Mensajes de Información

Nota: Esta tabla contiene las pruebas de caja negra realizadas al módulo de productos.

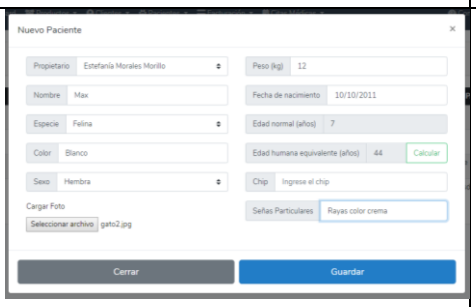
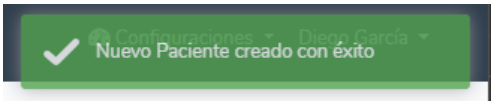
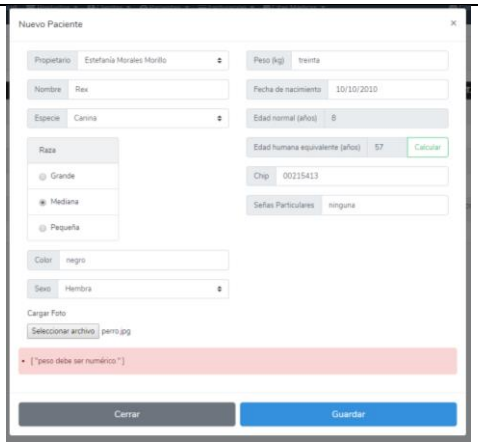
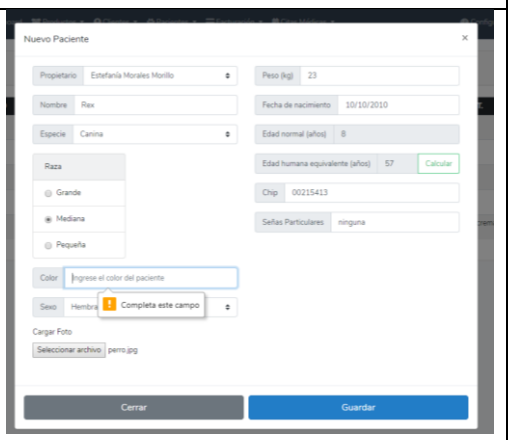
Tabla 42. Pruebas de Caja Negra del módulo Clientes

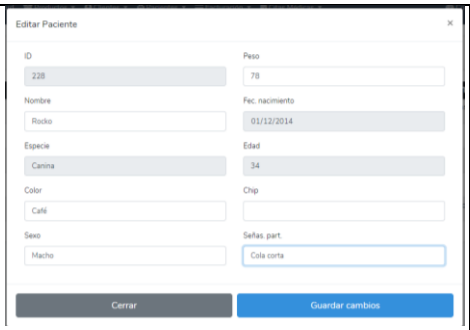
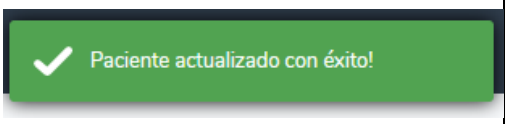
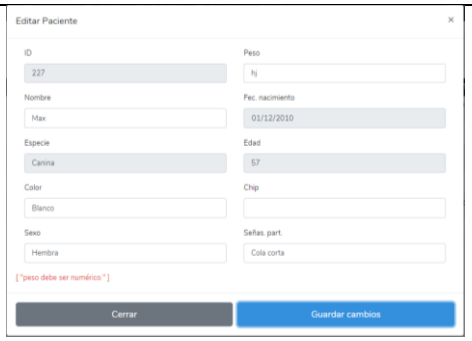
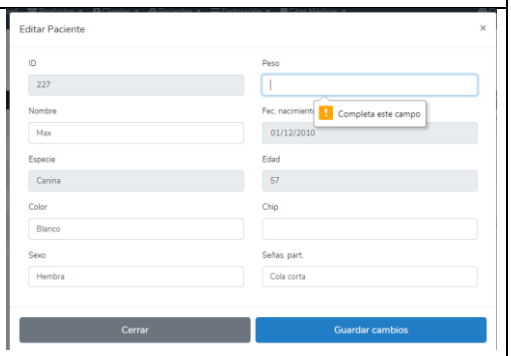
Func	**Tipo de entrada**	**Salida**
Crear Cliente	 Entrada Valida	 Mensaje de información

Crear Cliente	Entrada Incompleta	 Mensajes de información
Crear Cliente	Entrada No Valida	 Mensajes de alerta
Editar Cliente	 Entrada valida	 Mensaje de información
Editar Cliente	Entrada no Valida	

Nota: Esta tabla contiene las pruebas de caja negra realizadas al módulo de Clientes.

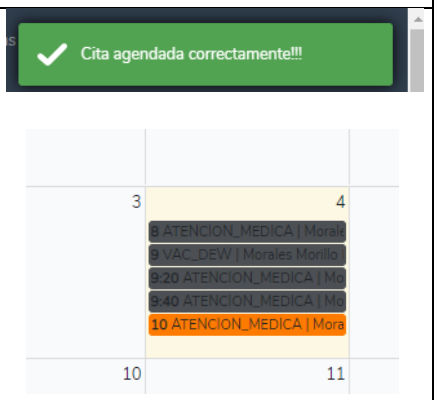
Tabla 43. Pruebas de Caja Negra del módulo Pacientes

Func	**Tipo de entrada**	**Salida**
Crear Pacientes	 Entrada valida	 Mensaje de información
Crear Pacientes	Entrada no valida	 Mensajes de alerta
Crear Paciente	Entrada Incompleta	 Mensajes de Información

Editar Paciente	 Entrada valida	 Mensaje de información
Editar Paciente	Entrada no valida	 Mensaje de alerta
Editar Paciente	Entrada Incompleta	 Mensajes de información

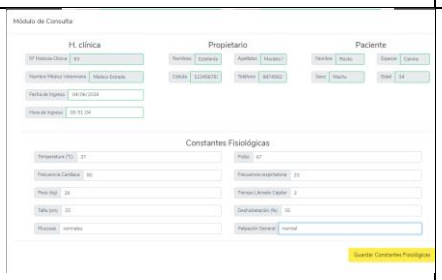
Nota: Esta tabla contiene las pruebas de caja negra realizadas al módulo de Pacientes.

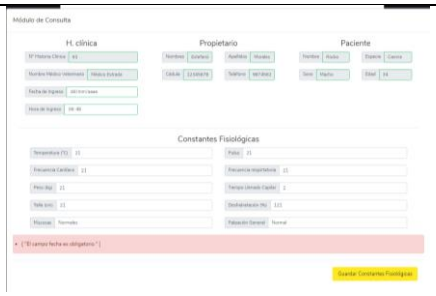
Tabla 44. Pruebas de Caja Negra del módulo Citas Médicas

Func.	Tipo de entrada	Salida
Crear Cita Medica	 Entrada valida	 Mensajes de información

Nota: Esta tabla contiene las pruebas de caja negra realizadas al módulo de Citas Médicas.

Tabla 45. Pruebas de Caja Negra del módulo de Consulta Médica

Func.	Tipo de entrada	Salida
Crear Consulta	 Entrada valida	 Mensaje de información
Crear Consulta	Entrada Incompleta	 Mensaje de información

Crear Consulta	Entrada No valida	 Mensajes de alerta

Nota: Esta tabla contiene las pruebas de caja negra realizadas al módulo de Consulta Médica.

4.1.2 Pruebas de Rendimiento.

Para realizar estas pruebas y las siguientes se ha utilizado el software Apache JMeter, en este software se ha configurado las pruebas de rendimiento evaluando la latencia de las peticiones HTTP frente a un número definido de concurrentes, inicialmente serán 100 luego 500 y finalmente 1000 hilos, cada uno de ellos en 1 segundo, de los resultados arrojados se comparará el rendimiento de cada uno.

4.1.2.1 Peticiones HTTP 100/1.

En la figura 17 se muestra la configuración que se realizó para la prueba de rendimiento, estableciendo el valor de Threads en “100” y en Period “1” lo que significa que se enviaran al servidor “100” peticiones HTTP en “1” segundo.

Configuración de la prueba de Rendimiento 100 hilos/1 segundo

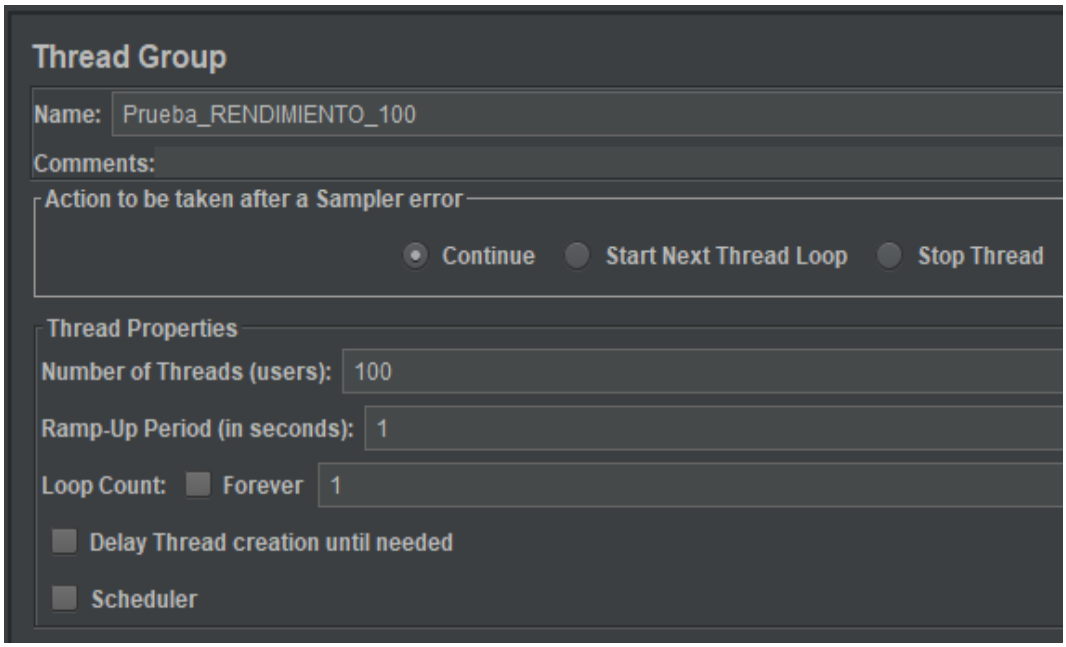


Figura: 17. Configuración de la prueba de Rendimiento de 100 hilos / 1 segundo
Elaborado por: Diego García Moya

En la siguiente figura vemos el resultado obtenido por el "Listener" Ver Resultados en árbol, las "100" peticiones fueron exitosas y podemos ver la respuesta del "Response data" que contiene el código html de la página del login.

Prueba de Rendimiento 100 hilos/1 segundo (View Results Tree)

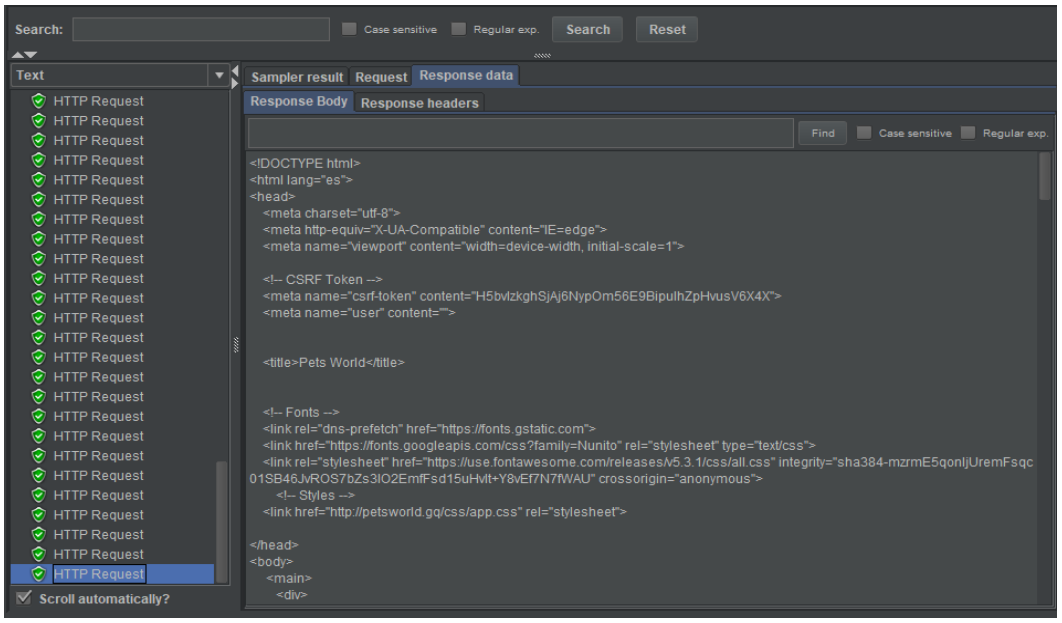
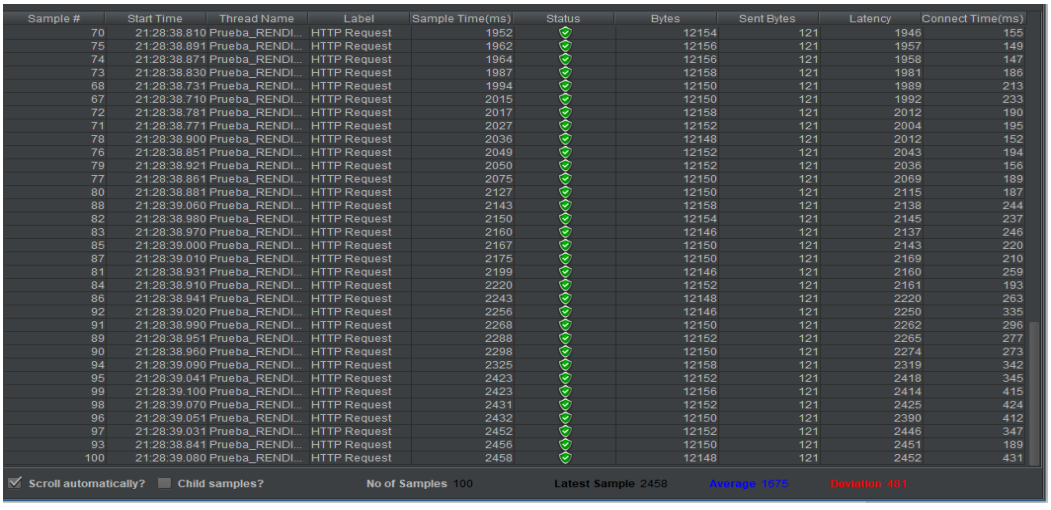


Figura: 18. Prueba de Rendimiento de 100 hilos / 1 segundo (View Results Tree)
Elaborado por: Diego García Moya

A continuación, vemos el resultado del Listener “Ver resultados en tabla”, el promedio de latencia es el dato más importante 1675 milisegundos.

Prueba de Rendimiento 100 hilos/1 segundo (View Results in table)



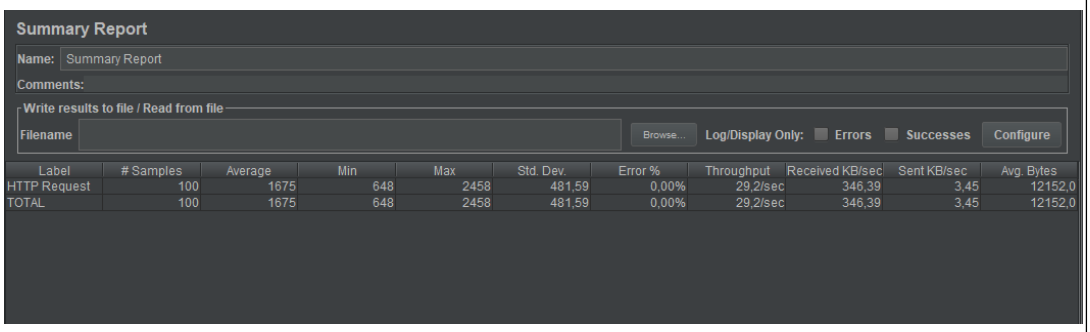
Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
70	21:28:38.810	Prueba_RENDI...	HTTP Request	1952		12154	121	1946	155
75	21:28:38.891	Prueba_RENDI...	HTTP Request	1962		12156	121	1957	149
74	21:28:38.871	Prueba_RENDI...	HTTP Request	1964		12156	121	1958	147
73	21:28:38.830	Prueba_RENDI...	HTTP Request	1987		12158	121	1981	186
68	21:28:38.731	Prueba_RENDI...	HTTP Request	1994		12150	121	1989	213
67	21:28:38.710	Prueba_RENDI...	HTTP Request	2015		12150	121	1992	233
72	21:28:38.781	Prueba_RENDI...	HTTP Request	2017		12158	121	2012	190
71	21:28:38.771	Prueba_RENDI...	HTTP Request	2027		12152	121	2004	195
78	21:28:38.900	Prueba_RENDI...	HTTP Request	2036		12148	121	2012	152
76	21:28:38.851	Prueba_RENDI...	HTTP Request	2049		12152	121	2043	194
79	21:28:38.921	Prueba_RENDI...	HTTP Request	2050		12152	121	2036	156
77	21:28:38.861	Prueba_RENDI...	HTTP Request	2075		12150	121	2069	189
80	21:28:38.881	Prueba_RENDI...	HTTP Request	2127		12150	121	2115	187
88	21:28:39.060	Prueba_RENDI...	HTTP Request	2143		12158	121	2138	244
82	21:28:38.980	Prueba_RENDI...	HTTP Request	2150		12154	121	2145	237
83	21:28:38.970	Prueba_RENDI...	HTTP Request	2160		12146	121	2137	246
85	21:28:39.000	Prueba_RENDI...	HTTP Request	2167		12150	121	2143	220
87	21:28:39.010	Prueba_RENDI...	HTTP Request	2175		12150	121	2169	210
81	21:28:38.931	Prueba_RENDI...	HTTP Request	2199		12146	121	2160	259
84	21:28:38.910	Prueba_RENDI...	HTTP Request	2220		12152	121	2161	193
86	21:28:38.941	Prueba_RENDI...	HTTP Request	2243		12148	121	2220	263
92	21:28:39.020	Prueba_RENDI...	HTTP Request	2256		12146	121	2250	335
91	21:28:38.990	Prueba_RENDI...	HTTP Request	2268		12150	121	2262	296
89	21:28:38.951	Prueba_RENDI...	HTTP Request	2288		12152	121	2265	277
90	21:28:38.960	Prueba_RENDI...	HTTP Request	2298		12150	121	2274	273
94	21:28:39.090	Prueba_RENDI...	HTTP Request	2325		12158	121	2319	342
95	21:28:39.041	Prueba_RENDI...	HTTP Request	2423		12152	121	2418	345
99	21:28:39.100	Prueba_RENDI...	HTTP Request	2423		12156	121	2414	415
98	21:28:39.070	Prueba_RENDI...	HTTP Request	2431		12152	121	2425	424
96	21:28:39.051	Prueba_RENDI...	HTTP Request	2432		12150	121	2390	412
97	21:28:39.031	Prueba_RENDI...	HTTP Request	2452		12152	121	2446	347
93	21:28:38.841	Prueba_RENDI...	HTTP Request	2456		12150	121	2451	189
100	21:28:39.080	Prueba_RENDI...	HTTP Request	2458		12148	121	2452	431

Scroll automatically? Child samples? No of Samples 100 Latest Sample 2458 Average 1675 Deviation 481

Figura: 19. Prueba de Rendimiento de 100 hilos / 1 segundo (View Results in table)
Elaborado por: Diego García Moya

Finalmente vemos un resumen de reporte con los valores que compararemos más adelante con las siguientes fases del Plan de Pruebas.

Prueba de Rendimiento 100 hilos/1 segundo (Summary Report)



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	100	1675	648	2458	481,59	0,00%	29,2/sec	346,39	3,45	12152,0
TOTAL	100	1675	648	2458	481,59	0,00%	29,2/sec	346,39	3,45	12152,0

Figura: 20. Prueba de Rendimiento de 100 hilos / 1 segundo (Summary Report)
Elaborado por: Diego García Moya

4.1.2.2 Peticiones HTTP 500/1.

Para la siguiente fase del Plan de Pruebas se ha configurado 500 peticiones HTTP que realizaran peticiones en el periodo de 1 segundo, a continuación, la configuración establecida.

Configuración de la prueba de Rendimiento 500 hilos/1 segundo

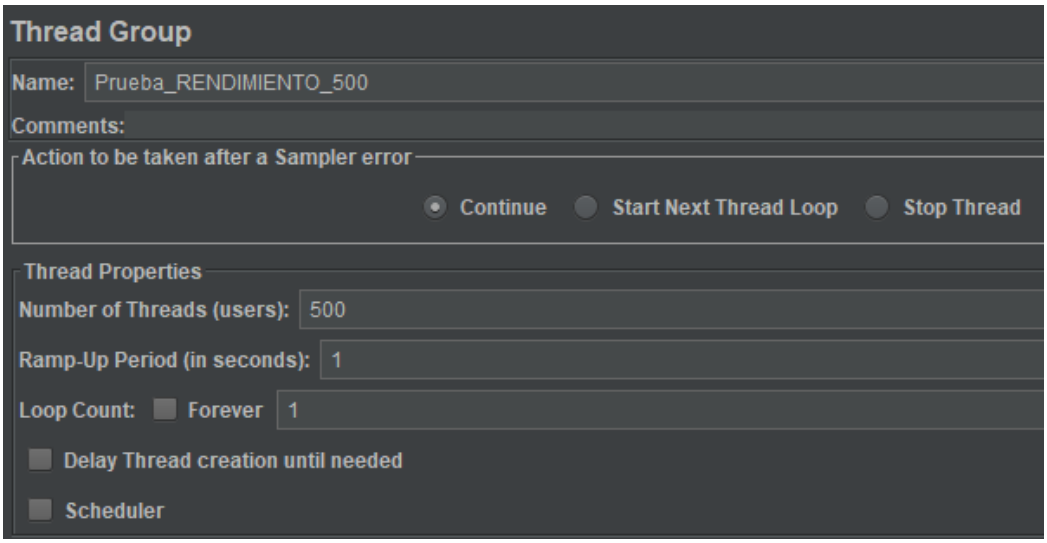


Figura: 21. Configuración de la prueba de Rendimiento de 500 hilos / 1 segundo
Elaborado por: Diego García Moya

La siguiente figura muestra los resultados obtenidos en el Listener “Ver Resultados en árbol”, 486 peticiones fueron resueltas correctamente por el servidor, se observa además la respuesta a la petición por parte del servidor, es el HTML que se usó para la página de autenticación.

Prueba de Rendimiento 500 hilos/1 segundo (View Results Tree)

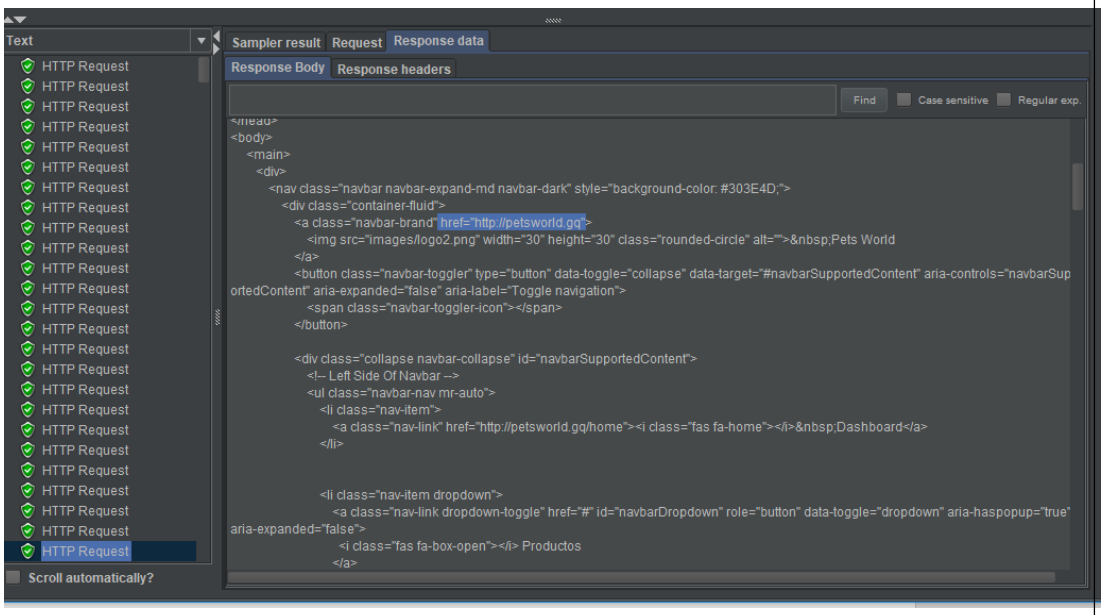
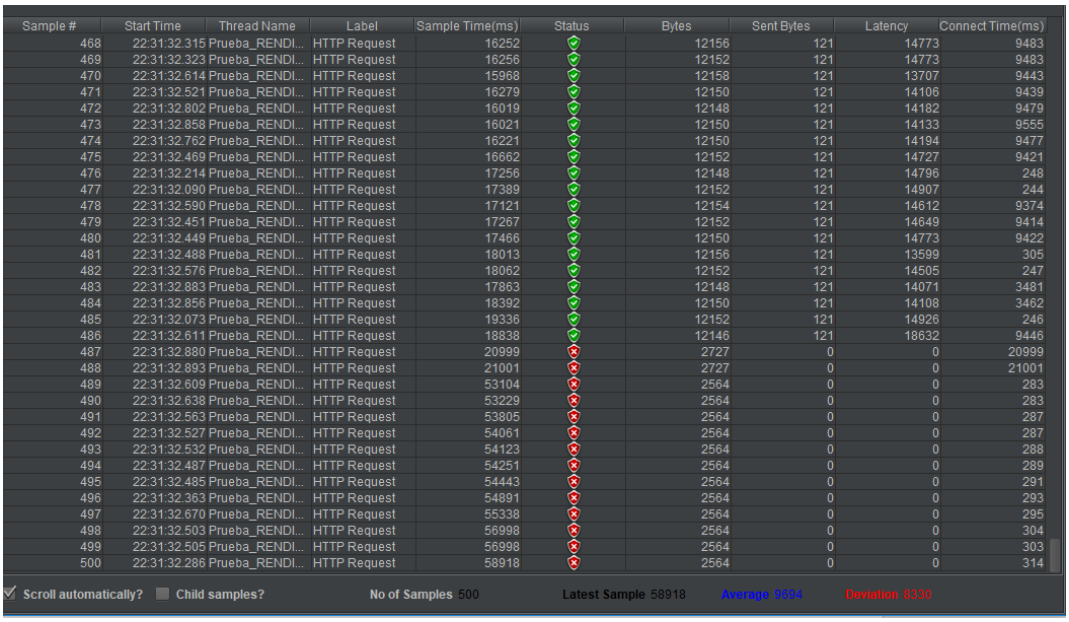


Figura: 22. Prueba de Rendimiento de 500 hilos / 1 segundo (View Results Tree)
Elaborado por: Diego García Moya

A continuación, en la siguiente figura se observa el promedio de la Latencia de 500 peticiones de las cuales solo 486 fueron atendidas correctamente por el servidor.

Prueba de Rendimiento 500 hilos/1 segundo (View Results in table)



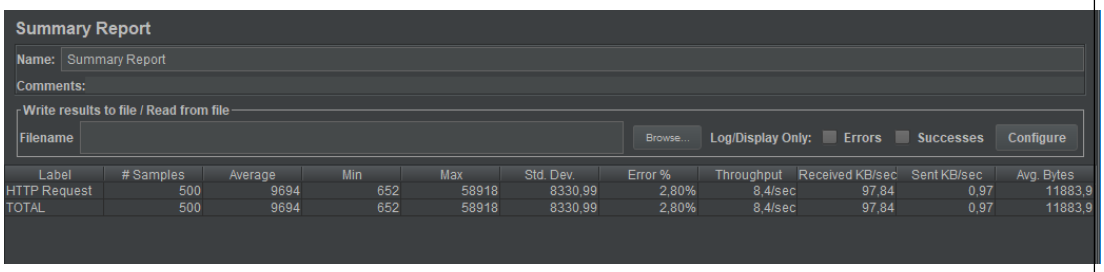
Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
468	22:31:32.315	Prueba_RENDI...	HTTP Request	16252	✓	12156	121	14773	9483
469	22:31:32.323	Prueba_RENDI...	HTTP Request	16256	✓	12152	121	14773	9483
470	22:31:32.614	Prueba_RENDI...	HTTP Request	15968	✓	12158	121	13707	9443
471	22:31:32.521	Prueba_RENDI...	HTTP Request	16279	✓	12150	121	14106	9439
472	22:31:32.802	Prueba_RENDI...	HTTP Request	16019	✓	12148	121	14182	9479
473	22:31:32.858	Prueba_RENDI...	HTTP Request	16021	✓	12150	121	14133	9555
474	22:31:32.762	Prueba_RENDI...	HTTP Request	16221	✓	12150	121	14194	9477
475	22:31:32.469	Prueba_RENDI...	HTTP Request	16662	✓	12152	121	14727	9421
476	22:31:32.214	Prueba_RENDI...	HTTP Request	17256	✓	12148	121	14796	248
477	22:31:32.090	Prueba_RENDI...	HTTP Request	17389	✓	12152	121	14907	244
478	22:31:32.590	Prueba_RENDI...	HTTP Request	17121	✓	12154	121	14612	9374
479	22:31:32.451	Prueba_RENDI...	HTTP Request	17267	✓	12152	121	14649	9414
480	22:31:32.449	Prueba_RENDI...	HTTP Request	17466	✓	12150	121	14773	9422
481	22:31:32.488	Prueba_RENDI...	HTTP Request	18013	✓	12156	121	13599	305
482	22:31:32.576	Prueba_RENDI...	HTTP Request	18062	✓	12152	121	14505	247
483	22:31:32.883	Prueba_RENDI...	HTTP Request	17863	✓	12148	121	14071	3481
484	22:31:32.856	Prueba_RENDI...	HTTP Request	18392	✓	12150	121	14108	3462
485	22:31:32.073	Prueba_RENDI...	HTTP Request	19336	✓	12152	121	14926	246
486	22:31:32.611	Prueba_RENDI...	HTTP Request	18838	✓	12146	121	18632	9446
487	22:31:32.880	Prueba_RENDI...	HTTP Request	20999	✗	2727	0	0	20999
488	22:31:32.893	Prueba_RENDI...	HTTP Request	21001	✗	2727	0	0	21001
489	22:31:32.609	Prueba_RENDI...	HTTP Request	53104	✗	2564	0	0	283
490	22:31:32.638	Prueba_RENDI...	HTTP Request	53229	✗	2564	0	0	283
491	22:31:32.563	Prueba_RENDI...	HTTP Request	53805	✗	2564	0	0	287
492	22:31:32.527	Prueba_RENDI...	HTTP Request	54061	✗	2564	0	0	287
493	22:31:32.532	Prueba_RENDI...	HTTP Request	54123	✗	2564	0	0	288
494	22:31:32.487	Prueba_RENDI...	HTTP Request	54251	✗	2564	0	0	289
495	22:31:32.485	Prueba_RENDI...	HTTP Request	54443	✗	2564	0	0	291
496	22:31:32.363	Prueba_RENDI...	HTTP Request	54891	✗	2564	0	0	293
497	22:31:32.670	Prueba_RENDI...	HTTP Request	55338	✗	2564	0	0	295
498	22:31:32.503	Prueba_RENDI...	HTTP Request	56998	✗	2564	0	0	304
499	22:31:32.505	Prueba_RENDI...	HTTP Request	56998	✗	2564	0	0	303
500	22:31:32.286	Prueba_RENDI...	HTTP Request	58918	✗	2564	0	0	314

Scroll automatically? Child samples? No of Samples 500 Latest Sample 58918 Average 9694 Deviation 8330

Figura: 23. Prueba de Rendimiento de 500 hilos / 1 segundo (View Results in table)
Elaborado por: Diego García Moya

Finalmente tenemos el Summary Report que resume los resultados de la prueba. Vemos el promedio de latencia en 9694 la latencia mínima en 652 y la máxima en 58918 mili segundos.

Prueba de Rendimiento 500 hilos/1 segundo (Summary Report)



Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename Browse... Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	500	9694	652	58918	8330,99	2,80%	8,4/sec	97,84	0,97	11883,9
TOTAL	500	9694	652	58918	8330,99	2,80%	8,4/sec	97,84	0,97	11883,9

Figura: 24. Prueba de Rendimiento de 500 hilos / 1 segundo (Summary Report)
Elaborado por: Diego García Moya

4.1.2.3 Peticiones HTTP 1000/1.

La configuración usada se muestra en la siguiente figura.

Configuración de la prueba de Rendimiento 1000 hilos/1 segundo

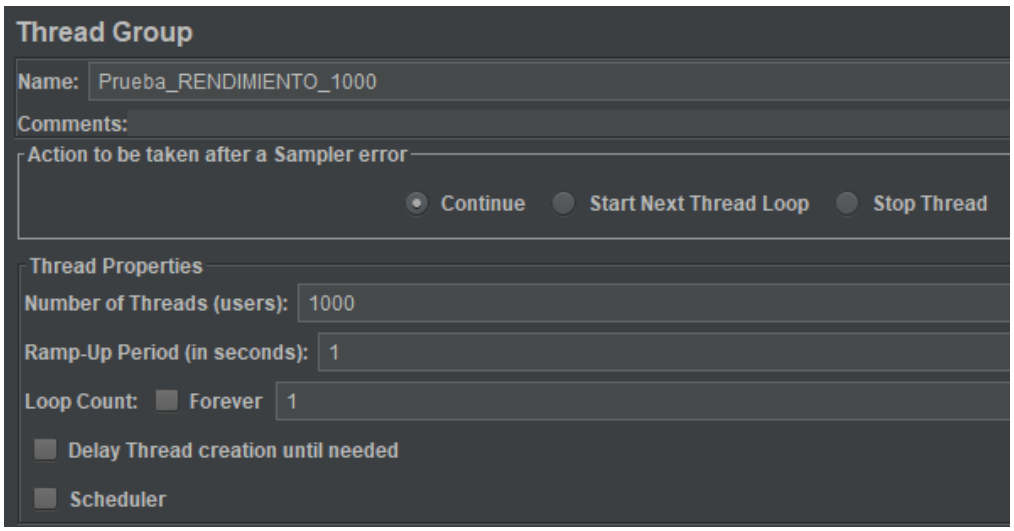


Figura: 25. Configuración de la prueba de Rendimiento de 1000 hilos / 1 segundo
Elaborado por: Diego García Moya

En la Figura siguiente podemos observar que varias peticiones empiezan a fallar debido a que han alcanzado el “Time out”.

Prueba de Rendimiento 1000 hilos/1 segundo (View Results in table)

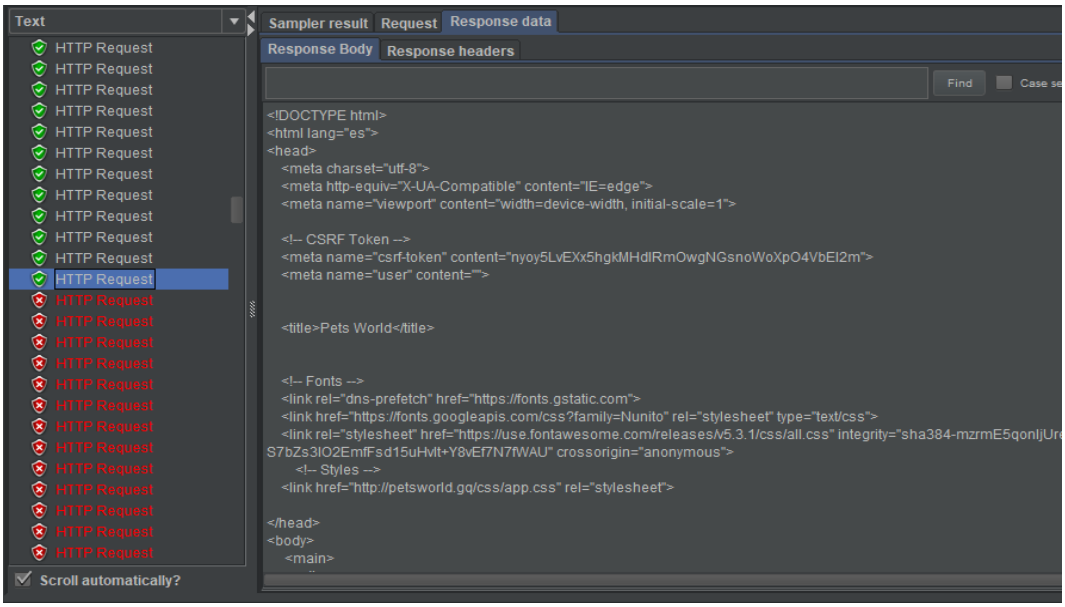


Figura: 26. Prueba de Rendimiento de 1000 hilos / 1 segundo (View Results in table)
Elaborado por: Diego García Moya

A continuación, la figura muestra que 669 peticiones fueron procesadas correctamente, a partir de ese punto empiezan a fallar.

Prueba de Rendimiento 1000 hilos/1 segundo (View Results in table)

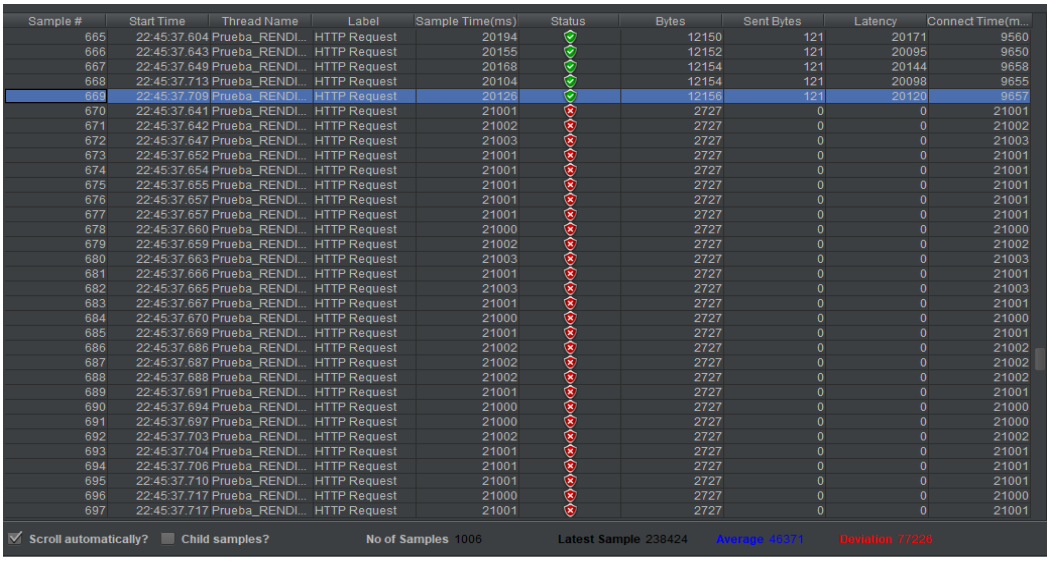
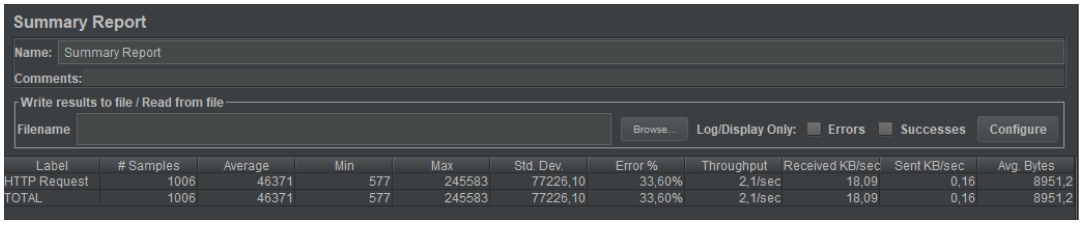


Figura: 27. Prueba de Rendimiento de 1000 hilos / 1 segundo (View Results in table)
Elaborado por: Diego García Moya

En el Summary Report se aprecia el promedio de latencia, latencia mínima y máxima,

Prueba de Rendimiento 1000 hilos/1 segundo (Summary Report)



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	1006	46371	577	245583	77226,10	33,60%	2,1/sec	18,09	0,16	8951,2
TOTAL	1006	46371	577	245583	77226,10	33,60%	2,1/sec	18,09	0,16	8951,2

Figura: 28. Prueba de Rendimiento de 1000 hilos / 1 segundo (Summary Report)
Elaborado por: Diego García Moya

Como conclusión podemos decir que el servidor web está preparado para entregar óptimo rendimiento hasta máximo 450 peticiones HTTP simultaneas lo cual lo hace con una latencia de 9.964 mili segundos.

4.1.3 Pruebas de Carga.

A diferencia del Test de Rendimiento que mide la capacidad de un servidor para responder a una cantidad de peticiones en un espacio de tiempo corto, el test de Carga permite medir la capacidad del servidor de mantener un funcionamiento constante y con las mismas prestaciones por un tiempo prolongado.

A continuación, se contrastan tres pruebas de Carga, la primera enviará 100 peticiones HTTP durante 20 segundos, la segunda 500 peticiones HTTP por 20 segundos y finalmente la tercera prueba enviará 1000 peticiones HTTP por 20 segundos también.

Comparación de Pruebas de Carga

100 peticiones / 20 segundos

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	100	396	329	717	59,89	0,00%	5,0/sec	58,94	0,59	12151,1
TOTAL	100	396	329	717	59,89	0,00%	5,0/sec	58,94	0,59	12151,1

500 peticiones 20 segundos

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	500	399	329	861	46,45	0,00%	24,6/sec	292,17	2,91	12152,0
TOTAL	500	399	329	861	46,45	0,00%	24,6/sec	292,17	2,91	12152,0

1000 peticiones / 20 segundos

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	1000	3061	366	10924	1541,42	0,00%	40,9/sec	485,25	4,83	12151,7
TOTAL	1000	3061	366	10924	1541,42	0,00%	40,9/sec	485,25	4,83	12151,7

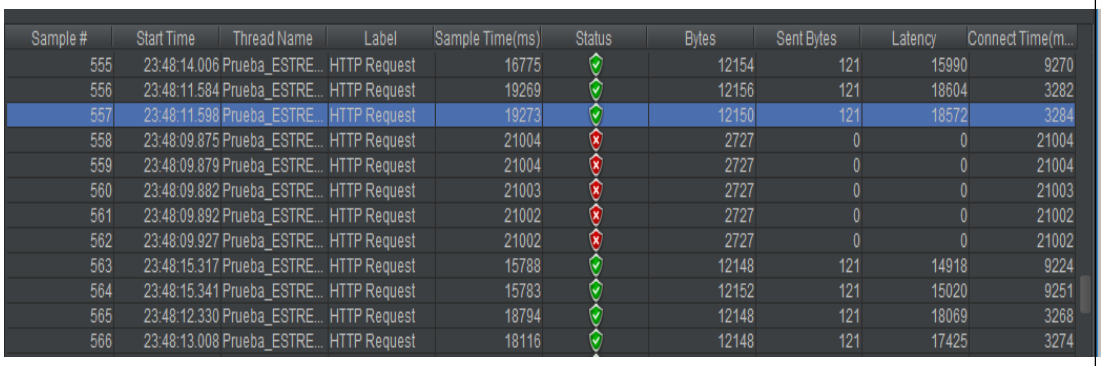
Figura: 29. Comparación de las Pruebas de Carga
Elaborado por: Diego García Moya

Se puede concluir que el test de Carga con mejores resultados es cuando se realiza la prueba de 500 peticiones, 20 segundos ya que los promedios de los tiempos máximos y mínimos de latencia son estables y se mantienen incluso con el primer test de 100 peticiones, 20 segundos.

4.1.4 Pruebas de Estrés.

Para realizar esta prueba se configurará un hilo que enviara 2000 peticiones en un espacio de 5 segundos, el objetivo es detectar según los resultados a partir de cuantas peticiones el servidor empieza a falla y finalmente hasta quedar of line.

Prueba de Estrés (1)

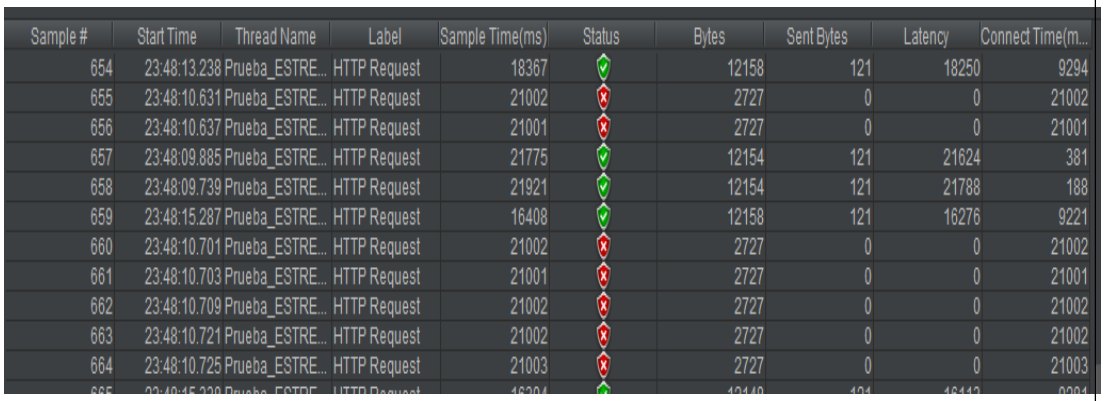


Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(m...
555	23:48:14.006	Prueba_ESTRE...	HTTP Request	16775		12154	121	15990	9270
556	23:48:11.584	Prueba_ESTRE...	HTTP Request	19269		12156	121	18604	3282
557	23:48:11.598	Prueba_ESTRE...	HTTP Request	19273		12150	121	18572	3284
558	23:48:09.875	Prueba_ESTRE...	HTTP Request	21004		2727	0	0	21004
559	23:48:09.879	Prueba_ESTRE...	HTTP Request	21004		2727	0	0	21004
560	23:48:09.882	Prueba_ESTRE...	HTTP Request	21003		2727	0	0	21003
561	23:48:09.892	Prueba_ESTRE...	HTTP Request	21002		2727	0	0	21002
562	23:48:09.927	Prueba_ESTRE...	HTTP Request	21002		2727	0	0	21002
563	23:48:15.317	Prueba_ESTRE...	HTTP Request	15788		12148	121	14918	9224
564	23:48:15.341	Prueba_ESTRE...	HTTP Request	15783		12152	121	15020	9251
565	23:48:12.330	Prueba_ESTRE...	HTTP Request	18794		12148	121	18069	3268
566	23:48:13.008	Prueba_ESTRE...	HTTP Request	18116		12148	121	17425	3274

Figura: 30. Prueba Estrés (1)
Elaborado por: Diego García Moya

En la figura 30 se puede ver que en el sample 557 empieza a fallar el servidor.

Prueba de Estrés (2)

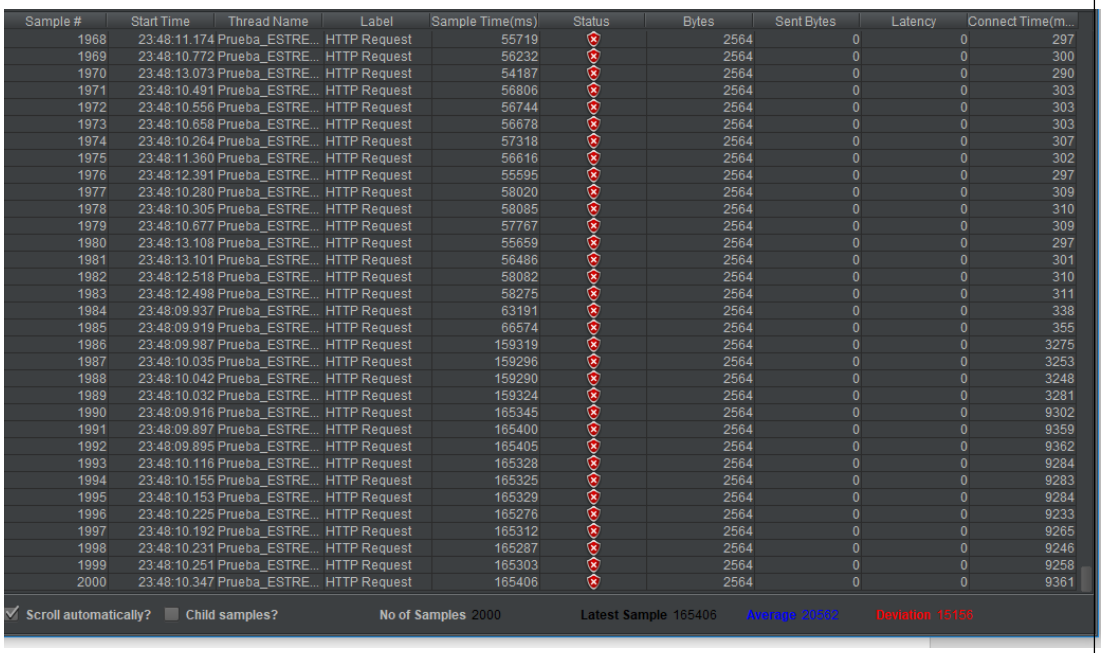


Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(m...
654	23:48:13.238	Prueba_ESTRE...	HTTP Request	18367		12158	121	18250	9294
655	23:48:10.631	Prueba_ESTRE...	HTTP Request	21002		2727	0	0	21002
656	23:48:10.637	Prueba_ESTRE...	HTTP Request	21001		2727	0	0	21001
657	23:48:09.885	Prueba_ESTRE...	HTTP Request	21775		12154	121	21624	381
658	23:48:09.739	Prueba_ESTRE...	HTTP Request	21921		12154	121	21788	188
659	23:48:15.287	Prueba_ESTRE...	HTTP Request	16408		12158	121	16276	9221
660	23:48:10.701	Prueba_ESTRE...	HTTP Request	21002		2727	0	0	21002
661	23:48:10.703	Prueba_ESTRE...	HTTP Request	21001		2727	0	0	21001
662	23:48:10.709	Prueba_ESTRE...	HTTP Request	21002		2727	0	0	21002
663	23:48:10.721	Prueba_ESTRE...	HTTP Request	21002		2727	0	0	21002
664	23:48:10.725	Prueba_ESTRE...	HTTP Request	21003		2727	0	0	21003

Figura: 31. Prueba Estrés (2)
Elaborado por: Diego García Moya

En la figura 31 el servidor nuevamente empieza a fallar con más frecuencia a partir de la petición 654.

Prueba de Estrés (3)



Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(m...
1968	23:48:11.174	Prueba_ESTRE...	HTTP Request	55719		2564	0	0	297
1969	23:48:10.772	Prueba_ESTRE...	HTTP Request	56232		2564	0	0	300
1970	23:48:13.073	Prueba_ESTRE...	HTTP Request	54187		2564	0	0	290
1971	23:48:10.491	Prueba_ESTRE...	HTTP Request	56806		2564	0	0	303
1972	23:48:10.556	Prueba_ESTRE...	HTTP Request	56744		2564	0	0	303
1973	23:48:10.658	Prueba_ESTRE...	HTTP Request	56678		2564	0	0	303
1974	23:48:10.264	Prueba_ESTRE...	HTTP Request	57318		2564	0	0	307
1975	23:48:11.360	Prueba_ESTRE...	HTTP Request	56616		2564	0	0	302
1976	23:48:12.391	Prueba_ESTRE...	HTTP Request	55595		2564	0	0	297
1977	23:48:10.280	Prueba_ESTRE...	HTTP Request	58020		2564	0	0	309
1978	23:48:10.305	Prueba_ESTRE...	HTTP Request	58085		2564	0	0	310
1979	23:48:10.677	Prueba_ESTRE...	HTTP Request	57767		2564	0	0	309
1980	23:48:13.108	Prueba_ESTRE...	HTTP Request	55659		2564	0	0	297
1981	23:48:13.101	Prueba_ESTRE...	HTTP Request	56486		2564	0	0	301
1982	23:48:12.518	Prueba_ESTRE...	HTTP Request	58082		2564	0	0	310
1983	23:48:12.498	Prueba_ESTRE...	HTTP Request	58275		2564	0	0	311
1984	23:48:09.937	Prueba_ESTRE...	HTTP Request	63191		2564	0	0	338
1985	23:48:09.919	Prueba_ESTRE...	HTTP Request	66574		2564	0	0	355
1986	23:48:09.987	Prueba_ESTRE...	HTTP Request	159319		2564	0	0	3275
1987	23:48:10.035	Prueba_ESTRE...	HTTP Request	159296		2564	0	0	3253
1988	23:48:10.042	Prueba_ESTRE...	HTTP Request	159290		2564	0	0	3248
1989	23:48:10.032	Prueba_ESTRE...	HTTP Request	159324		2564	0	0	3281
1990	23:48:09.916	Prueba_ESTRE...	HTTP Request	165345		2564	0	0	9302
1991	23:48:09.897	Prueba_ESTRE...	HTTP Request	165400		2564	0	0	9359
1992	23:48:09.895	Prueba_ESTRE...	HTTP Request	165405		2564	0	0	9362
1993	23:48:10.116	Prueba_ESTRE...	HTTP Request	165328		2564	0	0	9284
1994	23:48:10.155	Prueba_ESTRE...	HTTP Request	165325		2564	0	0	9283
1995	23:48:10.153	Prueba_ESTRE...	HTTP Request	165329		2564	0	0	9284
1996	23:48:10.225	Prueba_ESTRE...	HTTP Request	165276		2564	0	0	9233
1997	23:48:10.192	Prueba_ESTRE...	HTTP Request	165312		2564	0	0	9265
1998	23:48:10.231	Prueba_ESTRE...	HTTP Request	165287		2564	0	0	9246
1999	23:48:10.251	Prueba_ESTRE...	HTTP Request	165303		2564	0	0	9258
2000	23:48:10.347	Prueba_ESTRE...	HTTP Request	165406		2564	0	0	9361

Scroll automatically? Child samples? No of Samples 2000 Latest Sample 165406 Average 20562 Deviation 15156

Figura: 32. Prueba Estrés (3)
Elaborado por: Diego García Moya

Finalmente, el servidor deja de responder, como conclusión el servidor alcanza el umbral superior de su tolerancia en la petición 557, a partir de este punto el servidor falla hasta colapsar.

CONCLUSIONES

- Como se mencionó en el capítulo de introducción, el agendamiento de citas manual generaba muchos inconvenientes, finalmente mediante la integración de “Axios” para la captura de peticiones XMLHttpRequest del lado del cliente y “Eloquent” para realizar operaciones CRUD sobre la base de datos se ha logrado solucionar la duplicidad de citas médicas y por consiguiente se mejoró el proceso de agendamiento de citas médicas.

- Por los resultados obtenidos en las pruebas realizadas concluimos que la infraestructura que soporta al sistema, mantiene un umbral máximo de 450 usuarios accediendo al mismo tiempo, con lo que asegura que los usuarios tengan acceso permanente al sistema.

- El “ORM de Laravel” brinda la capacidad de convertir las entidades de la base de datos en modelos escritos en lenguaje PHP permite acceder a la información con las funciones y características de la Programación Orientada a Objetos, por consecuencia se pueden generar consultas complejas a la base de datos para mantener control en tiempo real de las existencias en bodega.

- La implementación de la arquitectura “MVC” (Modelo Vista Controlador) facilita el crecimiento e integración de nuevas funcionalidades a futuro, tales como, facturación electrónica, monitoreo por cámaras web a sala de las jaulas de mascotas. Como consecuencia de esta implementación, el sistema es cien por ciento escalable.

RECOMENDACIONES

- Se recomienda la creación de un módulo que permita el acceso de visitantes desde internet, para registrarse y agendar citas médicas sin necesidad de hacerlo personalmente, esto agilizaría el proceso de agendamiento de manera considerable.

- Se recomienda Integrar un módulo para “Backup” de la información en la base de datos de manera periódica, para preservar la información de posibles fallos.

- Se recomienda la creación de un módulo de seguridad, que administraría las cámaras de vigilancia instaladas en la institución, permitiendo acceso desde internet en cualquier momento al personal de la veterinaria que cuenten con los permisos y roles adecuados.

GLOSARIO DE TÉRMINOS

TRANSPILAR: Es la generación de código fuente en un lenguaje de programación a partir de un código fuente original, este código final tendrá el mismo comportamiento que el original.

ASSETS: Recursos que son indispensables para el funcionamiento de un sistema, como archivos de configuración.

ECMASCRIPT: "Es una especificación de Lenguaje de Programación publicada por ECMA Internacional." (International, 2019)

NPM: Por sus siglas, Node Package Manager es un gestor de paquetes de NodeJS.

ORM: Object Relational Maping.

SQL: Structured Query Language.

SPA: Single Page Application.

DOM: Document Object Model.

AJAX: Asynchronous Javascript and XML.

SYBASE: Empresa de desarrollo de software, fundada en 1984 y adquirida por SAP en 2010.

VCPU: Virtual Central Process Unit

LISTA DE REFERENCIAS

Bibliografía

Cobo, Á. (2015). *Tecnología para el desarrollo de aplicaciones web.* Madrid : Ediciones Díaz de Santos.

DevJoker. (s.f.). *DevJoker*. Obtenido de http://www.devjoker.com/contenidos/catss/525/Patron-MVC-Modelo-Vista-Controlador.aspx

Gutierrez, E. (2009). *Conceptos básicos y avanzados (bibliotecas Prototype y Script.aculo.us).* Barcelona: Ediciones ENI.

International, E. (20 de Marzo de 2019). *ECMA International*. Obtenido de ECMA International: https://www.ecma-international.org

Ken Schwaber, J. S. (2017). La Guía Scrum. En J. S. Ken Schwaber, *La Guía Scrum* (pág. 15).

Node.JS. (01 de Marzo de 2019). *Node JS*. Obtenido de Node JS: https://nodejs.org/es/

Rafael Camps Paré, L. A. (2015). *Bases de Datos Formación de Postgrado*. Cataluña: Eureca Media, SL. Obtenido de https://www.uoc.edu/masters/oficiales/img/913.pdf

S.A.S, E. (2014). *Programación Avanzada en Java.* Armenia, Quindio.

Schwaber, K. (2017). *La Guía de Scrum.*

Synspace. (2019). *TIOBE*. Obtenido de www.tiobe.com