UNIVERSIDAD NACIONAL DE CHIMBORAZO

FACULTAD DE INGENIERÍA

CARRERA DE SISTEMAS Y COMPUTACIÓN

Sistema web para la gestión de ventas del taller de reparación y mantenimiento automotriz Heredia utilizando el framework Mean Stack

Trabajo de Titulación para optar al título de Ingeniero en Sistemas y Computación

Autor:

Santiago Rafael Heredia Sayay

Tutora:

Ing. Miryan Estela Narváez Vilema

Riobamba, Ecuador. 2022

DERECHO DE AUTORÍA

Yo, Santiago Rafael Heredia Sayay, con cédula de ciudadanía 0605016179, autor del trabajo de investigación titulado: **Sistema web para la gestión de ventas del taller de reparación y mantenimiento automotriz Heredia utilizando el framework Mean Stack**, certifico que la producción, ideas, opiniones, criterios, contenidos y conclusiones expuestas son de mí exclusiva responsabilidad.

Asimismo, cedo a la Universidad Nacional de Chimborazo, en forma no exclusiva, los derechos para su uso, comunicación pública, distribución, divulgación y/o reproducción total o parcial, por medio físico o digital; en esta cesión se entiende que el cesionario no podrá obtener beneficios económicos. La posible reclamación de terceros respecto de los derechos de autor (a) de la obra referida, será de mi entera responsabilidad; librando a la Universidad Nacional de Chimborazo de posibles obligaciones.

En Riobamba, a la fecha de su presentación.

_____

Santiago Rafael Heredia Sayay

C.I: **0605016179**

ACTA FAVORABLE - INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN CARRERAS NO VIGENTES

En la Ciudad de Riobamba, a los 10 días del mes de noviembre de 2022, luego de haber revisado el Informe Final del Trabajo de Investigación presentado por el estudiante **SANTIAGO RAFAEL HEREDIA SAYAY** con CC: **0605016179**, de la carrera **INGENIERÍA EN SISTEMAS Y COMPUTACIÓN** y dando cumplimiento a los criterios metodológicos exigidos, se emite el **ACTA FAVORABLE DEL INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN** titulado **"SISTEMA WEB PARA LA GESTIÓN DE VENTAS DEL TALLER DE REPARACIÓN Y MANTENIMIENTO AUTOMOTRIZ HEREDIA UTILIZANDO EL FRAMEWORK MEAN STACK"**, por lo tanto se autoriza la presentación del mismo para los trámites pertinentes.

Firmado electrónicamente por:
MIRYAN ESTELA
NARVAEZ VILEMA

Ing. Miryan Estela Narváez Vilema
TUTORA

DICTAMEN FAVORABLE DEL TUTOR Y MIEMBROS DE TRIBUNAL

Quienes suscribimos, catedráticos designados Tutor y Miembros del Tribunal de Grado para la evaluación del trabajo de investigación **Sistema web para la gestión de ventas del taller de reparación y mantenimiento automotriz Heredia utilizando el framework Mean Stack**, presentado por Santiago Rafael Heredia Sayay con cédula de identidad número 0605016179, certificamos que recomendamos la APROBACIÓN de este con fines de titulación. Previamente se ha asesorado durante el desarrollo, revisado y evaluado el trabajo de investigación escrito y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable firmamos, en Riobamba a la fecha de su presentación.

Ing. Pamela Buñay, Mgs.
PRESIDENTE DEL TRIBUNAL DE GRADO _____

Ing. Lady Espinoza
MIEMBRO DEL TRIBUNAL DE GRADO _____

Ing. Ana Congacha
MIEMBRO DEL TRIBUNAL DE GRADO _____

Ing. Estela Narváez
TUTOR _____

CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

Quienes suscribimos, catedráticos designados Miembros del Tribunal de Grado para la evaluación del trabajo de investigación Sistema web para la gestión de ventas del Taller de Reparación y Mantenimiento Automotriz Heredia utilizando el framework Mean Stack por Santiago Rafael Heredia Sayay, con cédula de identidad número 0605016179, bajo la tutoría de Ing. Miryan Estela Narváez Vilema; certificamos que recomendamos la APROBACIÓN de este con fines de titulación. Previamente se ha evaluado el trabajo de investigación y escuchada la sustentación por parte de su autor; no teniendo más nada que observar

De conformidad a la normativa aplicable firmamos, en Riobamba

Presidente del Tribunal de Grado
Ing. Pamela Buñay, Mgs.

Firma

Miembro del Tribunal de Grado
Ing. Ana Congacha, Mgs.

Firma

Miembro del Tribunal de Grado
Ing. Lady Espinoza, Mgs.

Firma

C E R T I F I C A C I Ó N

Que, **HEREDIA SAYAY SANTIAGO RAFAEL** con CC: **0605016179**, estudiante de la Carrera **SISTEMAS Y COMPUTACIÓN**, **NO VIGENTE,** Facultad de **INGENIERÍA**; ha trabajado bajo mi tutoría el trabajo de investigación titulado **"SISTEMA WEB PARA LA GESTIÓN DE VENTAS DEL TALLER DE REPARACIÓN Y MANTENIMIENTO AUTOMOTRIZ HEREDIA UTILIZANDO EL FRAMEWORK MEAN STACK"**, cumple con el **0%,** de acuerdo al reporte del sistema Anti plagio **URKUND**, porcentaje aceptado de acuerdo a la reglamentación institucional, por consiguiente autorizo continuar con el proceso.

Riobamba, 10 de noviembre de año

Firmado electrónicamente por:
MIRYAN ESTELA
NARVAEZ VILEMA

Ing. Miryan Narváez
TUTORA TRABAJO DE INVESTIGACIÓN

DEDICATORIA

Dedico este proyecto a mi padre, a la memoria de mi madre, a mi hija y a mis hermanas, quienes fueron las personas que me inspiraron a ser la persona que ahora soy.

Santiago Heredia

AGRADECIMIENTO

Agradezco a mis padres por haberme brindado el apoyo para lograr cumplir con esta meta, a mis hermanas por formar parte integral de mi vida y participes en mis logros personales.

A mi tutora Ing. Miryan Narváez por el tiempo y apoyo en la dirección del proyecto.

A los docentes que tuve a lo largo de mi carrera, por la formación académica y moral indispensable para mi desarrollo profesional.

A mis compañeros y amigos de la universidad, ya que de ellos aprendí el valor del trabajo en equipo.

Santiago Heredia

ÍNDICE GENERAL

CAPÍTULO I 16
1. INTRODUCCION 16
1.1 PLANTEAMIENTO DEL PROBLEMA 17
1.1.1 Planteamiento del problema 17
1.1.2 Justificación 17
1.2 OBJETIVOS 18
1.2.1 Objetivo general 18
1.2.2 Objetivos específicos 18
CAPITULO II 19
2. MARCO TEORICO 19
2.1 ¿Qué es un Framework? 19
2.1.1 Ventajas y desventajas de un Framework en el desarrollo web 19
2.1.2 Tipos de Frameworks 20
2.2 Mean Stack 21
2.2.1 Arquitectura de Mean Stack 22
2.3 API Rest 22
2.3.1 REST o RESTful (Representational State Transfer) 23
2.4 Protocolo HTTP 23
2.5 JavaScript 24
2.6 MongoDB 24
2.6.1 Transacciones Maestro-Detalle en MongoDB 24
2.7 Express.js 25
2.8 Angular.js 26
2.9 NodeJS 26
2.10 Postman 27
2.11 Metodologías ágiles para el desarrollo de software 27
2.11.1 El Manifiesto Ágil 27
2.11.2 Metodología Kanban 28
2.12 Herramienta de rendimiento JMeter 28
CAPITULO III 29
3. METODOLOGIA 29
3.1 Tipo y diseño de la investigación 29
3.2 Población y muestra 29
3.3 Identificación de variables 30

3.4 Operacionalización de variables 30
3.5 Metodología de desarrollo 30
3.5.1 Tablero Kanban Primera Etapa 30
3.5.1.1 Análisis de requerimientos 31
3.5.2 Tablero Kanban Segunda Etapa 33
3.5.2.1 Diseño de la interfaz 33
3.5.2.2 Diseño de la arquitectura 36
3.5.3 Tablero Kanban Tercera Etapa 38
3.5.3.1 Desarrollo del Backend 39
3.5.3.2 Desarrollo del Frontend 45
3.5.4 Tablero Kanban Cuarta Etapa 64
3.5.4.1 Pruebas de rendimiento del sistema web 65
3.5.5 Tablero Kanban Quinta Etapa 66
3.5.6 Tablero Kanban Sexta Etapa (Final) 67
CAPÍTULO IV 69
4 RESULTADOS Y DISCUSIÓN 69
CAPITULO V 75
5 CONCLUSIONES y RECOMENDACIONES 75
CONCLUSIONES 75
RECOMENDACIONES 75
BIBLIOGRAFÍA 77

ÍNDICE DE TABLAS

Tabla 1: Ventajas y desventajas de los frameworks 19
Tabla 2: Comparativa entre métodos SQL y HTTP 23
Tabla 3: Operacionalización con variables 30
Tabla 4: Actividades planificadas en la Primer Etapa 30
Tabla 5: Requerimientos funcionales del sistema web 32
Tabla 6: Requerimientos no funcionales del sistema web 33
Tabla 7: Actividades planificadas en la Segunda Etapa 33
Tabla 8: Actividades planificadas en la Tercera Etapa 38
Tabla 9: Herramientas para el desarrollo del sistema web 39
Tabla 10: Actividades planificadas en la cuarta etapa 64
Tabla 11: Actividades planificadas en la quinta etapa 66
Tabla 12: Actividades planificadas en la sexta etapa 67
Tabla 13: Prueba de rendimiento con 1 thread. 70
Tabla 14: Prueba de rendimiento con 10 threads. 70
Tabla 15: Prueba de rendimiento con 20 threads 71
Tabla 16: Prueba de rendimiento con 30 threads. 72

ÍNDICE DE FIGURAS

Figura 1: Arquitectura de un framework 20
Figura 2: Arquitectura de AJAX 21
Figura 3: Arquitectura de Mean Stack 22
Figura 4: Estructura de un Api Rest 22
Figura 5: Patrón maestro-detalle en MongoDB 25
Figura 6: Estructura de AngularJS 26
Figura 7: Tablero Kanban 28
Figura 8: Diagrama de casos de uso Sistema Web 31
Figura 9: Diseño del login 34
Figura 10: Diseño del módulo principal - Dashboard 34
Figura 11: Diseño del módulo de productos 35
Figura 12: Diseño del módulo de categorías 35
Figura 13: Diseño del módulo de clientes 35
Figura 14: Diseño del módulo de usuarios 36
Figura 15: Diseño del módulo de ventas 36
Figura 16: Arquitectura Model ViewModel Model 37
Figura 17: Modelado de datos a implementar en MongoDB 37
Figura 18: Creación del entorno de desarrollo del Backend 40
Figura 19: Instalación de paquetes y dependencias de Node 41
Figura 20: Conexión con MongoDB y Robo3T 41
Figura 21: Conexión del servidor con la base de datos 42
Figura 22: Estructura del modelo User 42
Figura 23: Controlador del modelo Ventas. 43
Figura 24: Ruta para la entidad Producto 44
Figura 25: Modulo app con las rutas del servidor 44
Figura 26: Insertar categoría desde Postman 45
Figura 27: Instalación de AngularJS y sus módulos 45
Figura 28: División del proyecto en Angular 47
Figura 29: Componentes de la carpeta Shared 47
Figura 30: Vista principal del sistema web 48
Figura 31: Maquetación con HTML del componente Login 49
Figura 32: Lógica de programación del componente Login 49
Figura 33: Servicios del componente Login 50
Figura 34: Vista principal del componente Login 50
Figura 35: Maquetación del Dashboard 51
Figura 36: Lógica de programación del componente Dashboard 51
Figura 37: Vista principal del componente Dashboard 52
Figura 38: Maquetación del componente Usuarios 53
Figura 39: Lógica de programación del componente Usuarios 53
Figura 40: Vista principal del componente Usuarios. 54
Figura 41: Maquetación del componente Clientes 54
Figura 42: Lógica de programación del componente Clientes 55
Figura 43: Vista principal del componente Clientes. 55
Figura 44: Maquetación del componente Proveedores. 56
Figura 45: Lógica de programación del componente Proveedores. 56

Figura 46: Vista principal del componente Proveedores. 57
Figura 47: Maquetación del componente Productos. 57
Figura 48: Lógica de programación del componente Productos 58
Figura 49: Vista principal del componente Productos. 58
Figura 50: Maquetación del componente Compras 59
Figura 51: Lógica de programación del componente Compras 60
Figura 52: Vista principal del componente Compras. 60
Figura 53: Maquetación del componente de Ventas. 61
Figura 54: Lógica de programación del componente Ventas 61
Figura 55: Vista principal del componente de Ventas 62
Figura 56: Lógica de programación del componente Reporte de compras 62
Figura 57: Vista del reporte de compras 63
Figura 58: Lógica de programación del componente Reporte de ventas 63
Figura 59: Vista del reporte de ventas 64
Figura 60: Captura de eventos con BlazeMeter 65
Figura 61: Eventos capturados por BlazeMeter 66
Figura 62: Prueba de tiempo de respuesta con 100 usuarios 69
Figura 63: Gráfica comparativa entre rendimiento y desviación con 1 usuario 70
Figura 64: Gráfica comparativa entre rendimiento y desviación con 10 usuarios 71
Figura 65: Gráfica comparativa entre rendimiento y desviación con 20 usuarios 72
Figura 66: Gráfica comparativa entre rendimiento y desviación con 30 usuarios 73

RESUMEN

Actualmente, usar un sistema de gestión administrativa se ha vuelto indispensable para toda empresa en crecimiento, mejora la credibilidad del negocio, genera ahorros y optimiza los procesos de manera sistemática. Existe mucha demanda de tecnologías para desarrollo de estos sistemas de gestión, sin embargo, debido a que el desarrollo de un sistema web implica recursos y personal se abre la oportunidad de introducción a un nuevo marco de desarrollo basado únicamente en el lenguaje JavaScript. Mean Stack es un framework de desarrollo para el cliente y servidor, utilizado para la creación de sistemas y aplicaciones web modernas, con un alto desempeño funcional, ágil procesamiento de información, escalabilidad y gran capacidad de almacenamiento.

Con este marco de desarrollo acompañado de una metodología ágil como Kanban se logró crear un sistema web administrativo enfocado al sector automotriz, el cual permite almacenar una gran cantidad de información de uso comercial en un servidor creado con MongoDB, Express y NodeJS. Para el consumo de esta información se crearon procesos y funciones en una interfaz de usuario desarrollada con AngularJS en su versión 10, la cual presenta la información obtenida del servidor en formato HTML.

Finalmente, el sistema web fue monitoreado para evaluar su rendimiento a través de pruebas de carga y estrés con las herramientas JMeter y BlazeMeter, las cuales demostraron tener un margen mínimo de error en procesamiento y almacenamiento de datos. Los resultados obtenidos se presentan a través de gráficas y árboles de resultados para una mejor interpretación.

Palabras clave: AngularJS, ExpressJS, Metodología Kanban, MongoDB, NodeJS.

SUMMARY

Nowadays, using a web management system has become indispensable for any growing company, it improves business credibility, generates savings and optimizes processes in a systematic way. There are many effective technologies for the development of web management systems, however, these technologies require human and economic resources that not all companies can afford. For this reason, a new development framework based on the JavaScript programming language is used. This framework does not require significant investments of money and reduces the number of development personnel. MEAN Stack is a framework used for the development of modern web applications and systems, which reload a single page, also known as SPA (Single Page Application), and with high storage capacity and scalability.

With this development framework, it was possible to create a management system focused on the automotive commercial sector using the MEAN Stack framework and the agile development methodology Kanban. This web system stores a large amount of commercial information in a server built with MongoDB, Express and NodeJS. The resulting data is represented in a graphical user interface developed with AngularJS.

Finally, the performance, response time and stress of the web system were evaluated with JMeter and BlazeMeter tools. The result of this evaluation showed that the web system developed with MEAN Stack can support massive data traffic. In addition, it has a 0% error margin in the resolution of commercial transactions such as: buying and selling products, updating data, user registration and inventory control.

Keywords: AngularJS, ExpressJS, Kanban Methodology, MongoDB, NodeJS

SANDRA LILIANA ABARCA GARCIA

Firmado digitalmente por SANDRA LILIANA ABARCA GARCIA
Fecha: 2022.11.16 15:56:04 -05'00'

CAPÍTULO I

1. INTRODUCCION

En 1995 se introdujo un nuevo método de interacción para páginas web desarrolladas con HTML para navegadores Netscape, consistía en brindar al usuario la capacidad de interactuar con una página web al enviar datos de tipo texto a un servidor y obtener respuestas, no obstante la validación de datos por parte del servidor suponía una espera excesiva para el usuario, tomando en cuenta que a principios de los 90s la velocidad de internet era de 28.8kb/s, un resultado más que suficiente considerando que en aquella década las páginas web se basaban solo en texto plano y en ocasiones imágenes de baja resolución. En la misma década y con la introducción de aplicaciones web más complejas se vio la necesidad de tener un lenguaje de programación por el lado del cliente, este debía ser capaz de validar formularios de datos más complejos antes de enviarlos al servidor con el fin de evitar el tiempo de espera agobiante, con esta idea si el usuario cometía errores al ingresar datos o si estos no estaban validados, ya no se tenía que esperar a que el servidor envíe una respuesta negativa. La evolución de JavaScript le ha permitido migrar hacia otras plataformas, ahora es posible manipular datos en el servidor utilizando como lenguaje de programación a JavaScript, y almacenarlos en una base de datos de MongoDB en formato JSON (Robledano, 2021).

En un mercado en constante evolución es necesario conocer nuevas tecnologías que permitan manipular tanto el lado del servidor como el lado del cliente de manera individual, JavaScript ofrece una solución que agrupa dentro de un marco o Stack de desarrollo a cuatro tecnologías basadas en este lenguaje, de estos se conocen: Mean (Mongo, Express, Angular, Node) y Mern (Mongo, Express, React, Node), ambos orientados al desarrollo de aplicaciones web. (UNIR, 2022). Las ventajas de utilizar estos marcos de desarrollo en la creación de aplicaciones web son básicamente el desarrollo separado de cliente y servidor, lo cual beneficia al trabajo organizado, la escalabilidad y la reducción de código innecesario. Gracias a una gran comunidad de soporte, Mean y Mern Stack se posicionan como una de las opciones más eficaces para el desarrollo de aplicaciones web progresivas, interactivas e intuitivas actualmente.

El desarrollo de sistemas web para apoyar las PyMES en Ecuador, muestra un notable crecimiento a raíz de que las microempresas desean mejorar sus servicios en cuanto a atención al cliente y competitividad en el mercado. Los servicios en alta demanda incluyen también la creación de aplicaciones delivery y administración de negocios digitales que incluyan pasarelas de pago online. Las tecnologías open source se encuentran en su máximo apogeo, cloud service, hosting, diseño web, y más tecnologías que apoyan la reactivación del comercio electrónico en el país. (Medrano, 2020)

El sector automotriz aporta un 2% en impuestos generados algo que económicamente hablando significa un valor positivo al desarrollo de la matriz productiva del país, además de generar valor mercantil a otros sectores. Según Padilla et al., (2020), los impuestos que

versan sobre el sector automotriz, al menos en Chimborazo, contribuyen al incremento socioeconómico provincial. Se toma en cuenta este precedente por la factibilidad de implementar sistemas de gestión administrativa que contribuyan al desarrollo y la satisfacción de necesidades, adecuando a las tecnologías de la información en este ámbito como apoyo para fomentar el avance tecnológico de este y otros sectores.

El desarrollo de este proyecto de investigación se centró en la creación de un sistema web de gestión de ventas, tomando como caso de estudio al taller de reparación y mantenimiento automotriz "Heredia". Se utilizó la metodología de desarrollo ágil Kanban, debido a su versatilidad en la integración e iteración de actividades, de igual manera se utilizó el framework Mean Stack para el control de la información comercial que se genere en el negocio, la información resultante se almacena en una base de datos no relacional con MongoDB y esta será consumida a través de una interfaz interactiva desarrollada con Angular.

1.1 PLANTEAMIENTO DEL PROBLEMA

1.1.1 Planteamiento del problema

La mayoría de las empresas comerciales buscan obtener mejores beneficios a través de las tecnologías de la información, los sistemas ERP, Enterprise Resource Planning, son los sistemas más utilizados en la actualidad para llevar un control empresarial y automatizar procesos en una sola plataforma. (Gallegos, 2016). Sin embargo, un sistema ERP supone un costo elevado, en especial para PyMES, lo que provoca que las empresas en crecimiento abandonen la idea de migrar a soluciones tecnológicas que pueden beneficiarla. Con este precedente surge el siguiente problema de investigación:

¿Por qué es importante desarrollar un sistema web con el framework MEAN Stack para la gestión de ventas?

1.1.2 Justificación

Actualmente, es indispensable para PyMES llevar un control de información administrativa que refleje el estado actual de sus movimientos, la información suele ser extensa y aumenta de manera exponencial con el tiempo. En el caso de estudio del taller de reparación y mantenimiento automotriz "Heredia", las actividades comerciales y administrativas son registradas de forma manual lo cual no garantiza un control exacto de la información, como se mencionó, la información crece exponencialmente y es necesario tenerla almacenada y disponible a todo momento para quienes hacen uso legítimo de ella.

Con la siguiente investigación se justifica el desarrollo de un sistema web con el framework Mean Stack, el cual almacena toda esta información en un servidor desarrollado con Mongo, Express y NodeJS. Para el consumo de esta información de

forma más interactiva se desarrollaron funciones, que permiten el consumo de la información a través de una interfaz gráfica desarrollada con Angular.

Finalmente, con este proyecto de investigación se propuso un nuevo marco de desarrollo de aplicaciones o sistemas web modernos acordes a las nuevas exigencias del mercado laboral, ya que actualmente se desea contar con desarrolladores que sepan del manejo individual de Backend y Frontend.

1.2 OBJETIVOS

1.2.1 Objetivo general

Implementar el sistema web para la gestión de ventas del taller de reparación y mantenimiento automotriz Heredia utilizando el framework Mean Stack.

1.2.2 Objetivos específicos

- Investigar el framework Mean Stack para el desarrollo de sistemas y páginas web dinámicas.
- Desarrollar el módulo de ventas del sistema web utilizando el framework MEAN Stack.
- Evaluar el rendimiento del sistema web utilizando la herramienta JMeter.

CAPITULO II

2. MARCO TEORICO

2.1 ¿Qué es un Framework?

Según (Ciceri, 2019), es un término relacionado a un subsistema o un marco de librerías con diferentes funcionalidades que se pueden acoplar en cualquier sistema estándar, entre otras funcionalidades, un framework aporta:

- Mejor organización en el código, cuya ventaja es una mejor refactorización y escalabilidad del software.
- Un framework está en constante actualización, ya que detrás de estos existe una comunidad de desarrolladores que mantienen su seguridad e integridad.
- Brindan confianza a los nuevos desarrolladores.
- Oportunidad de aportar con nuevas técnicas o métodos, un framework siempre está al alcance de cualquier entusiasta en el desarrollo de software.
- Generan en los desarrolladores la costumbre de implementar buenas prácticas de programación.

Para sintetizar los aportes mencionados, un framework es un entorno de trabajo sustentable para el desarrollo de software, que permite al programador evitar el excesivo uso de líneas de código para el desarrollo de un proyecto, lo que lo vuelve viable, escalable y legible en el tiempo.

2.1.1 Ventajas y desventajas de un Framework en el desarrollo web

En la tabla 1, se muestra las ventajas y desventajas de utilizar un framework para el desarrollo de sistemas web.

Tabla 1: *Ventajas y desventajas de los frameworks*

Ventajas	Desventajas
• Proporcionan un marco de trabajo • Es eficiente en la codificación, ya que reutiliza código. • Brinda agilidad y rapidez en el desarrollo de software. • Agilizan el uso de patrones de diseño, por ejemplo, MVC (Modelo Vista Controlador).	• Es indispensable conocer y superar la curva de aprendizaje que implica manejar un framework. • Las nuevas versiones casi siempre son inestables. • Ocupan un mayor espacio de trabajo. • Pueden existir aplicaciones pequeñas que no necesiten una sobre carga de código, añadir un

• Tienen soporte de varios programadores ya que la mayoría son de código libre. • Es adaptable con otras librerías. • Es mantenible.	framework solo las volvería más pesadas. • La gran variedad de frameworks hacen que sea difícil elegir el más apropiado para el desarrollo de software.

Fuente: (Bravo, 2018)

Elaborado por: *El autor, 2022*

2.1.2 Tipos de Frameworks

Según la autora (Muente, 2020), para el desarrollo web existe una gran variedad de frameworks que facilitan el desarrollo de software con pocos recursos, de los cuales los más importantes destacan:

Framework para aplicaciones web. – Los frameworks de este tipo se comportan de manera indiferente al lenguaje de programación que se emplea en el desarrollo web, sin embargo, no está por demás saber que se puede fácilmente migrar de un lenguaje de programación a otro sin tener que alterar la estructura del framework o del software.

Framework para aplicaciones en general. – Diseñados para complementar un sistema operativo existente, en el caso de Microsoft su framework .NET permite que los programadores reutilicen recursos existentes. Por lo general, este framework ya viene instalado en Microsoft. En la Figura 1, se aprecia la arquitectura del framework .NET de Microsoft. Su funcionamiento se basa en dos componentes: la biblioteca de clases y el CLS.

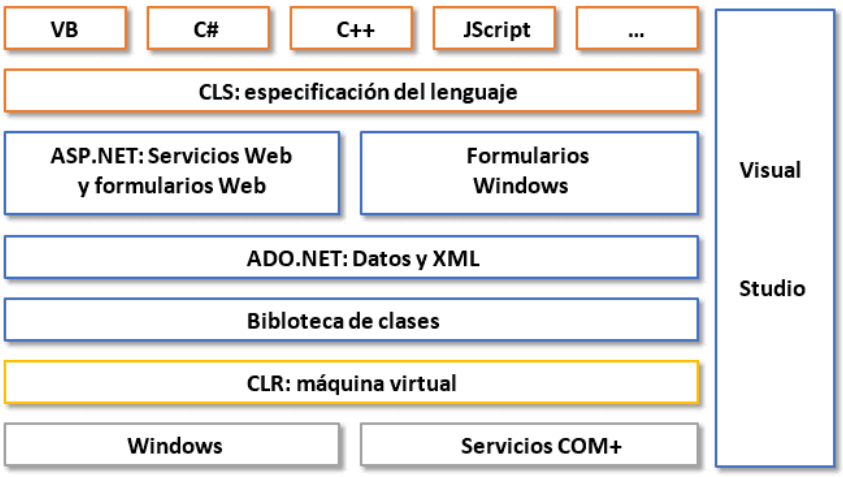

Figura 1: *Arquitectura de un framework*

Obtenido de: *https://www.glosarioit.com/GloImg/NETFramework.png*

Framework para tecnología AJAX. – AJAX (Asynchronous JavaScript And XML) permite la comunicación asíncrona con el servidor sin necesidad de recargar el DOM (Document Object Model). Para esta tecnología existen framework que utilizan otros frameworks que permiten la comunicación asíncrona, como Mean Stack. En la Figura 2, se observa el esquema de funcionamiento de AJAX, las peticiones que se realizan al servidor son resueltas en segundo plano, evitando recargas innecesarias de la página.

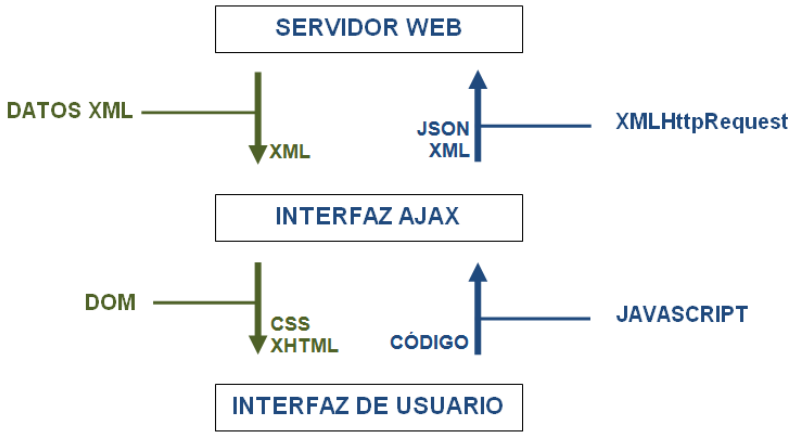

Figura 2: *Arquitectura de AJAX*

Obtenido de: *https://laredinfinita.wordpress.com/2014/04/22/esquema-de-ajax/*

Framework de gestión de contenidos. – También conocidos como CMF (Content Management Framework), permiten a los CMS (Content Management System) la facilidad de controlar con mejor desempeño el manejo multimedia de páginas o sitios web por parte de editores de contenido.

Framework de multimedia. – Estos framework permiten la gestión de contenido multimedia como: audio, fotos, o videoconferencias. Por lo general, este framework es más utilizado por empresas de marketing digital que manejen la creación de contenido de alta calidad.

2.2 Mean Stack

El termino MEAN es una abreviación del conjunto de subsistemas basados en el lenguaje de programación JavaScript. MEAN está compuesto por MongoDB, Express.js, AngularJS y NodeJS, las ventajas que brinda este conjunto son bastas, de las cuales destacan:

- Utilizar un solo lenguaje de programación para el desarrollo de software.
- A menudo, este tipo de subsistemas soportan una arquitectura MVC.
- Estructura los datos en forma de texto y los transmite de un sistema a otro gracias a JSON. (Mejía, Haviv, & Onodi, 2017)

2.2.1 Arquitectura de Mean Stack

Para el desarrollo de una arquitectura basada en MEAN Stack es importante tener un bloque de transferencia de estado representacional o API Rest, que nutre de información a una aplicación de página única SPA. Comúnmente, esta API Rest se desarrolla con MongoDB, Espress.js y NodeJS, y con la ayuda de una página web SPA desarrollada con AngularJS, se consigue un servicio rápido y eficiente de transferencia de información efectivo, rápido y fiable. (Holmes, 2019). En la Figura 3, se observa la arquitectura de Mean Stack y la resolución de peticiones entre cliente y servidor.

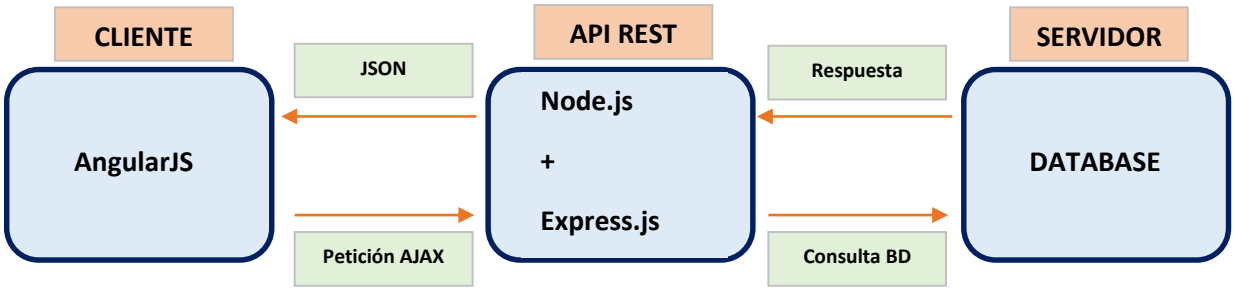

Figura 3: *Arquitectura de Mean Stack*

Elaborado por: *El autor, 2022*

2.3 API Rest

Según los autores (Chan, Chung , & Huang, 2019), entiéndase por API a la interfaz de programación de aplicaciones, que puede ser móvil o web, la cual sirve de puente de comunicación con la parte lógica de un software, esta herramienta constituye la base de la comunicación asíncrona entre Frontend y Backend. Uno de los más importantes beneficios de construir un API es la encapsulación de la información, al ser esta una interfaz estándar, puede ser consumida a través de cualquier terminal de Frontend, móvil o web. Con esta condición se construye una API con información confidencial, lista para satisfacer servicios externos a usuarios autorizados. En la Figura 4, se muestra un modelo de Api Rest, en donde las peticiones que envía el usuario tienen formato JSON, posteriormente se resuelven a través de los métodos HTTP hasta llegar al servidor quien enviará una respuesta a la petición inicial.

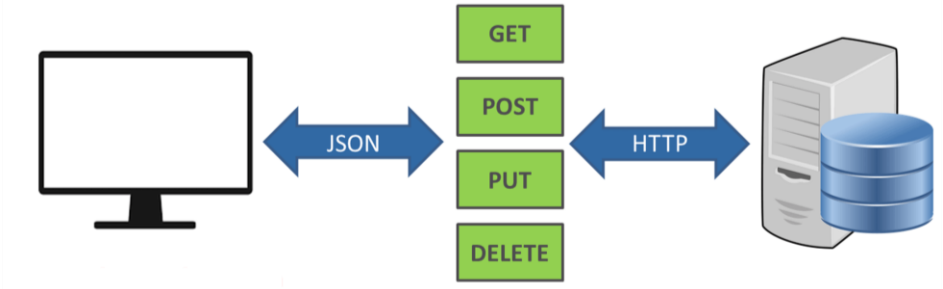

Figura 4: *Estructura de un Api Rest*

Obtenido de: *https://phpenthusiast.com/theme/assets/images/blog/what_is_rest_api.png*

2.3.1 REST o RESTful (Representational State Transfer)

Transferencia de estado representacional, REST por su definición en español, constituye una arquitectura de software basada en la web, capaz de soportar grandes volúmenes de tráfico, también destaca su escalabilidad y su eficiente uso de recursos de software.

Principios y restricciones de REST

El estilo arquitectónico de REST está formado de 5 principios/restricciones como:

- Existe independencia entre cliente y servicio, cualquier de las dos partes pueden ser reemplazadas, siempre y cuando exista una interfaz de por medio que permita la comunicación a través de peticiones.
- Las peticiones no tienen dependencia de ninguna de sus antecesoras, el estado de sesión se mantiene en el lado del cliente.
- Ya que es probable que las peticiones sean reutilizables, REST las almacena en cache para mejorar el rendimiento.
- La división de la parte lógica de los recursos permite una mejor manipulación de la memoria cache, esto resulta en varias peticiones a la vez, reduciendo recursos y aumentando la escalabilidad.
- La uniformidad de la interfaz intermedia permite un fácil desacoplamiento entre cliente y servidor.

2.4 Protocolo HTTP

Es la abreviatura de Protocolo de Transferencia de Hipertexto, en español, usado mundialmente y necesario de comprender para abordar de mejor manera la utilización de un API Rest. Todos los sitios, foros, revistas o cualquier tipo de repositorio virtual alojado en la web, necesita del prefijo HTTP para llegar a los usuarios a través de un buscador.Cuando interactúan el API con la Interfaz del cliente, lo hacen por medio de URLs, definidas con diferentes tipos de métodos HTTP, los cuales se presentan en la Tabla 2:

Tabla 2: *Comparativa entre métodos SQL y HTTP*

OPERACIÓN	SQL	HTTP	DDS
Create	INSERT	POST / PUT	Write
Read	SELECT	GET	Read / Take
Update	UPDATE	PUT / PATCH	Write
Delete	DELETE / DROP	DELETE	Dispose

Elaborado por: *El autor, 2022*

2.5 JavaScript

En 1995 la empresa tecnológica Netscape Communications (Mozilla Corp.) desarrolla un plugin para navegadores web cuya finalidad era de proporcionar un ambiente más dinámico a las páginas web de aquel entonces. Su creador Brendan Eich recibió críticas variadas por parte de desarrolladores convencionales quienes catalogaban a JavaScript como una mezcla entre el lenguaje de programación Java con otros lenguajes de programación basados en C. En el año 2008, el navegador Chrome con el motor de búsqueda V8 y de la mano de Google empezó a integrar nuevas funcionalidades que le permitían sostener aplicaciones web más complejas. Una de las características que cambio el punto de vista de los detractores de JavaScript fue el poder ejecutar aplicaciones en el navegador y no en el servidor personal del usuario, sin duda el mayor cambio que hasta el día de hoy se mantiene. En la actualidad JavaScript aparece dentro del top de lenguajes más populares para desarrollo web según GitHub, IEEE, TOIBE. (Salvaggio & Testa, 2019)

2.6 MongoDB

Es una base de datos de tipo NoSQL, que se orienta al uso de documentos, diferente a una base de datos relacional. Por lo tanto, según (Singh & Ahmad, 2019), NoSQL es un término que abarca la utilización de documentos con unidades de responsabilidad reducidas, esto quiere decir que, cada registro en una base de datos NoSQL se comporta como una unidad autónoma de información y, que permite la integración de datos relacionales para mayor beneficio y soluciones especializadas.

2.6.1 Transacciones Maestro-Detalle en MongoDB

Según el autor (Rosoff, 2018), las transacciones atómicas son las más comunes en las relaciones de tablas en una base de datos relacional, la normalización de tablas permite tener mayor acceso a consultas a través de sentencias SQL. Una base de datos en MongoDB está orientada a documentos y no a tablas, como es el caso de MySQL, SQL Server, sin embargo, MongoDB sabe compensarlo muy bien al incorporar el patrón Maestro-Detalle en sus documentos, consiguiendo de esta manera una consulta anidada. Para obtener un nivel de atomicidad similar a SQL, pero en MongoDB, se debe tomar en consideración lo siguiente:

- Se deben modelar los documentos como un objeto anidado enriquecido, esto quiere decir que, de igual manera que en las tablas se declara una llave foránea de tipo padre-hijo, en MongoDB se instancia un patrón de tipo clave-valor, el resultado es un documento con datos externos más enriquecido y flexible.

- Este tipo de patrón se debe utilizar cuando se requiere extraer datos externos de un documento a otro. Por lo general, en una base de datos NoSQL se puede añadir datos de forma intensificada, debido a que MongoDB no exige una normalización.
- Como se muestra en la Figura 5, el patrón maestro-detalle permite tener consultas anidadas al introducir a diferentes documentos la clave del documento Usuario en los documentos Contacto y Nivel, esta clave se ingresa como un objeto JSON del cual se obtiene todos los datos del documento principal.

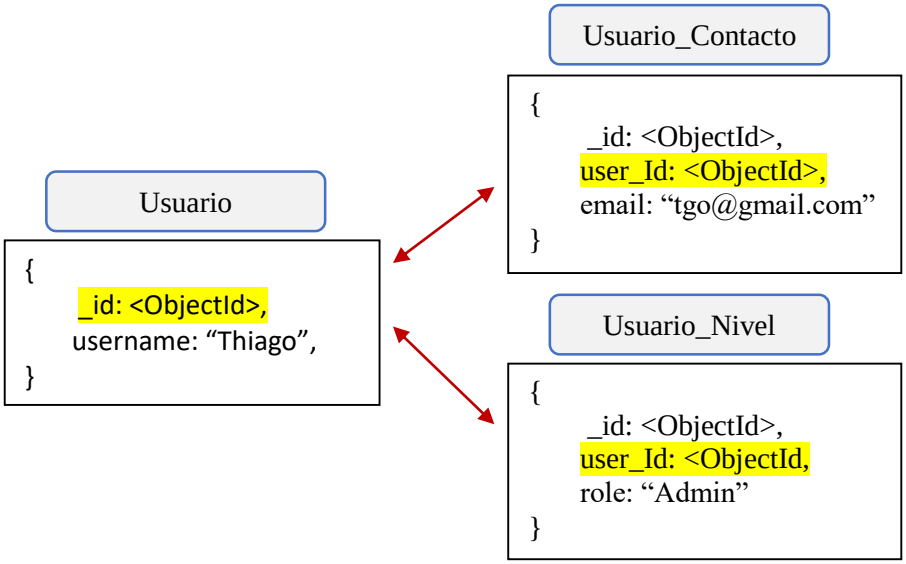

Figura 5: *Patrón maestro-detalle en MongoDB*

Elaborado por: *El autor, 2022*

2.7 Express.js

Según (Uzayr, 2022), el framework Express.js es mayoritariamente escogido por desarrolladores de software con NodeJS, por su parecido al framework Connect, Express es un marco de desarrollo minimalista que permite el enrutamiento de URLs, gestión de cookies, entre otros. Express permite el manejo renderizado de vistas, mediante el uso de motores de plantillas, lo que significa que, establece las mejores rutas en una aplicación web para conectarse con otras. Como plataforma de aplicaciones web, Express permite crear aplicaciones de una sola página, debido a que utiliza JavaScript permite una mejor organización del código. Mientras que el motor de enrutamiento responde a las solicitudes del cliente a través de los métodos GET y POST, se puede mapear de manera más eficiente una ruta diferente a cada petición.

2.8 Angular.js

De acuerdo con (Soni, 2018), Angular es un framework escrito en JavaScript, que sigue el patrón de diseño Modelo Vista Controlador. Es uno de los frameworks más populares actualmente, al ser de código libre permite mantener un nivel de alta escalabilidad. Una de sus mayores características es el desarrollo de aplicaciones de una sola página, lo que lo vuelve una tendencia en el mercado.

Las principales características que destacan a Angular como el framework preferido por desarrolladores son:

- Su estructura responde al patrón de diseño MVC, el más popular actualmente. MVC permite separar el modelo de datos de la lógica de usuario y de control, lo que lo vuelve fácil de mantener.
- A través de data binding y otras directivas de Angular, se puede vincular fácilmente las vistas HTML al modelo lógico del software.
- Con Angular se reduce la manipulación al DOM, lo que beneficia al desarrollo de aplicaciones móviles, web workers y de escritorio.
- A través de Karma, test-runner de Angular, realiza pruebas unitarias y automatizar tareas de las suites.

En la Figura 6, se muestra la estructura MVC de AngularJS y que a través del two-way databinding se puede interactuar entre la vista y el modelo de una aplicación web.

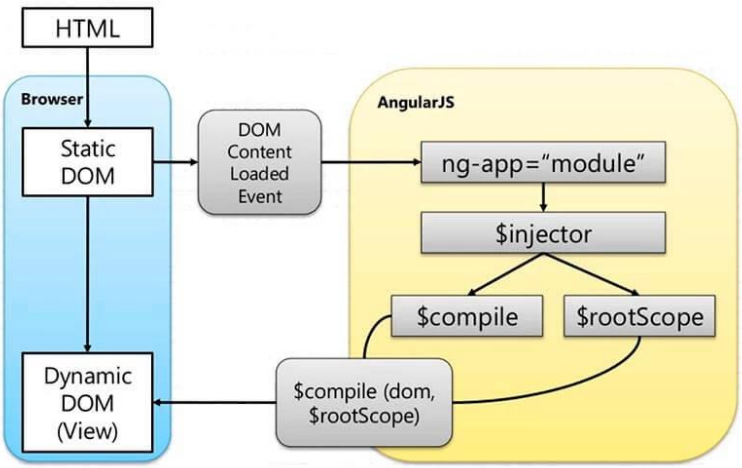

Figura 6: *Estructura de AngularJS*

Obtenido de:
https://dc722jrlp2zu8.cloudfront.net/media/cache/54/f5/54f52eff319ad720c5d01d97d71b89d9.webp

2.9 NodeJS

Es un entorno de desarrollo multiplataforma que se basa en el lenguaje de programación JavaScript, es de código abierto y orientado a eventos. Su motor de interpretación de JavaScript se denomina V8, y es de propiedad de Google. NodeJS se compila en tiempo de

ejecución, lo que permite optimizar funciones al ser llamadas, evitando cambios de contexto. Junto con Express.js crean una capa de abstracción de datos que permite la mejor interacción entre cliente y base de datos, adicionando servicios a los ya existente de manera nativa. NodeJS se basa en la comunicación asíncrona a través de eventos denominados Callbacks, estas funciones se llaman al finalizar una tarea, evitando de esta manera cualquier tipo de bloqueo (Puciarelli, 2020).

2.10 Postman

(Muradas, 2022) Postman es una herramienta tecnológica compatible con sistemas operativos Windows, Mac y Linux, cuyo objetivo es de testear las API Rest a través de peticiones HTTP tanto para el Backend como para el Frontend. Postman es imprescindible en el desarrollo de aplicaciones web, permite tener actualizadas las colecciones de datos mejorando el tiempo de respuesta del servicio, por otro lado, su comunidad de desarrollo es extensa por lo que siempre se encontrará documentación de respaldo en caso de existir inconvenientes en el trabajo.

2.11 Metodologías ágiles para el desarrollo de software

El término "ágil" fue empleado por un grupo de investigadores de la universidad de Utah, Estados Unidos, como parte de una investigación acerca de desarrollo de software que responda a cambios durante el proceso de ejecución y que se puedan dividir en tareas con el fin de delegar a otros miembros del equipo de desarrollo. Anteriormente los proyectos solían caracterizarse por ser demasiado rígidos y con excesiva documentación, este fue un enorme problema ya que no se podía establecer peticiones de cambio debido a la excesiva documentación por leer y redactar. Como resultado se resolvió crear un manual para facilitar el desarrollo ágil de software al cual se lo denomino "The Agile Alliance", teniendo como fin brindar un conjunto de conceptos y guías para agilizar el desarrollo de software.

2.11.1 El Manifiesto Ágil

Este manifiesto para el desarrollo ágil de software pone a consideración:

- Se debe crear primero un equipo de trabajo antes de empezar con un proyecto, esto facilita a los miembros acoplen sus capacidades en el entorno de desarrollo.
- El software resultante tiene más importancia que la documentación, los informes deben generarse en base a los resultados finales en cada fase del proyecto.
- Debe existir interacción entre equipo desarrollador y cliente, es muy común tener cambios inesperados de los cuales se debe estar siempre abierto a sugerencias (Canós , Letelier, & Penadés, 2003).

2.11.2 Metodología Kanban

Kanban es un modelo de representación visual de las etapas de desarrollo de un proyecto. Usado por primera vez por Taiichi Ohno (Toyota, JIT) y empleado en las cadenas de producción para control en la entrega a tiempo, almacenaje y fabricación de un producto. Kanban emplea un conjunto de reglas para el control como:

- Visualización de estados del proyecto.
- Determinación de los límites de las tareas en progreso.
- Medición del flujo del tiempo empleado en las tareas.

Esta metodología ayuda a un fácil acoplamiento de tareas, gracias a su diseño en columnas que separan los estados del proyecto, su ventaja principal es que siempre entrega información actual en cualquier estado en que se encuentre el proyecto. Además de favorecer su fácil implementación y acoplamiento a otros enfoques, organizar los equipos de trabajo y fácil delegación de tareas (Gaete et al., 2021). La Figura 7, se muestra el esquema estructural de un modelo Kanban separado por columnas en las cuales se representan cada uno de los estados del proyecto como: pendientes, en proceso y finalizada.

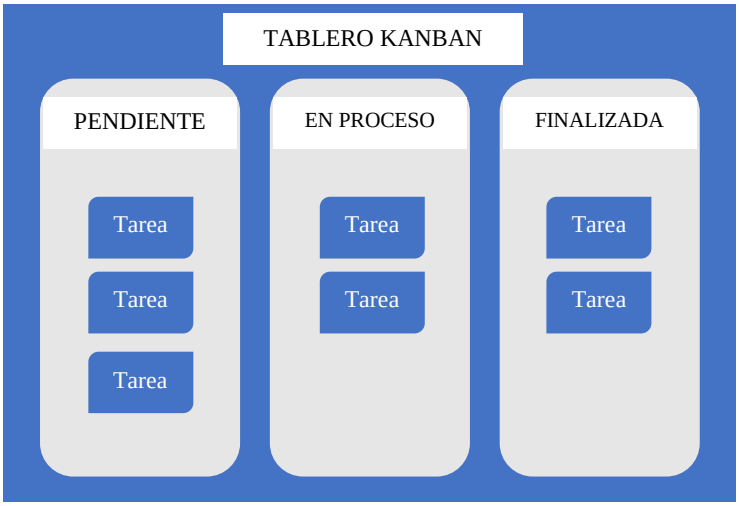

Figura 7: *Tablero Kanban*

Elaborado por: *El autor, 2022*

2.12 Herramienta de rendimiento JMeter

JMeter es una herramienta de evaluación de rendimiento y funcionalidad de código abierto desarrollado en el lenguaje java, orientado para aplicaciones web, aunque en la actualidad su funcionalidad se ha expandido a otros servicios tecnológicos para pruebas. Con esta herramienta también es posible medir el nivel de rendimiento de un servidor web al agregar cargas y peticiones a través de métodos HTTP.

CAPITULO III

3. METODOLOGIA

El desarrollo del sistema web para el control de ventas tiene un enfoque cuantitativo, porque se evalúo el rendimiento del sistema con la herramienta JMeter, ésta muestra diferentes métricas como: pruebas de carga, rendimiento y funcionalidad del sistema web, las cuales aportan al mejoramiento y escalabilidad del sistema.

3.1 Tipo y diseño de la investigación

Según el tipo de variable

- **Cuantitativa:** La investigación tienen un enfoque cuantitativo porque se evaluó el rendimiento del sistema web con la herramienta JMeter, cuya finalidad fue cuantificar y analizar los datos en términos de rendimiento, velocidad y pruebas de carga y estrés del sistema.

Según el objeto de estudio

- **Investigación de campo:** Se recopiló datos directamente sobre el estado actual de las finanzas del taller, talonario de facturas, libros de contabilidad, de igual manera se realizó un inventario en una hoja de cálculo con el fin de tener los datos necesarios para la creación del API Rest que consumirá el sistema web.
- **Investigación aplicada:** Es una investigación aplicada ya que aporta una solución tecnológica a un problema en específico, como es el caso de la gestión de ventas que a su vez resuelve el problema de la adquisición de mercadería. De igual manera se ve beneficiados el sector automotriz ya que se incursiona en una nueva técnica de administración de productos y mejoramiento en la calidad de atención al cliente.
-

Según la fuente de investigación

- **Investigación bibliográfica:** El proyecto recopiló información publicada en: libros, revistas científicas, foros, tesis de grado, páginas web, entre otras fuentes de información que permitieron llegar al desarrollo de un sistema web utilizando el framework Mean Stack.

3.2 Población y muestra

El sistema web será evaluado en base a su rendimiento con la herramienta JMeter, por tal motivo su población es de tipo infinita.

3.3 Identificación de variables

Variable independiente

- Sistema web utilizando el framework Mean Stack.

Variable dependiente

- Gestión de ventas del taller.

3.4 Operacionalización de variables

Tabla 3: *Operacionalización con variables*

Variable	Tipo	Definición Conceptual	Dimensión	Indicadores
Sistema web utilizando el framework MEAN Stack.	Independiente	MEAN Stack es un conjunto de subsistemas basado en JavaScript para el desarrollo de sistemas web SPA.	Sistema web para gestión de ventas.	• Módulo de ventas. • Número de usuarios. • Niveles de acceso.
Gestión de ventas del taller.	Dependiente	La gestión de ventas corresponde al conjunto de procesos contables que permiten el buen manejo de bienes de una empresa.	Gestión de ventas, reportes.	• Ventas netas. • Número de clientes. • Stock.

Elaborado por: *El autor, 2022*

3.5 Metodología de desarrollo

Para el desarrollo del sistema web de gestión de ventas se utilizó la metodología ágil Kanban, dividida en seis etapas. En cada etapa las tareas se dividen en 3 columnas denominadas: Pendiente, En Proceso y Finalizada, cada etapa muestra las actividades que se planificaron, ejecutaron y se dieron por finalizadas.

3.5.1 Tablero Kanban Primera Etapa

En la Tabla 4, se muestra las actividades planificadas en la primera etapa de desarrollo del sistema web.

Tabla 4: *Actividades planificadas en la Primer Etapa*

PENDIENTE	EN PROCESO	FINALIZADA
Revisar la documentación de TypeScript	Realizar la toma de requerimientos funcionales y no funcionales.	

Revisar la documentación de Node y Express	Realizar el diseño del logo empresarial	
Revisar la documentación de MongoDB	Realizar el diagrama de casos de uso del sistema web	
Revisar la documentación de Angular		

***Elaborado por**: El autor, 2022*

3.5.1.1 Análisis de requerimientos

Para el desarrollo del sistema web se realizó un estudio interno de las necesidades a cubrir. El sistema debe tener un control de acceso por medio de roles (administrador y vendedor), existen tareas administrativas restringidas para usuarios con el role predefinido como 'ADMIN', tales como: agregar, modificar y eliminar los datos dentro de los módulos que forman el sistema, estas tareas se gestionan a través de la interfaz gráfica de usuario. La única tarea que está restringida para el usuario con role predefinido como 'SELLER' o vendedor será la de agregar usuarios, producto y categorías al sistema web.En la Figura 8, se muestra el diagrama de casos de uso que representa la manera en que el sistema opera basado en los roles asignados a cada usuario.

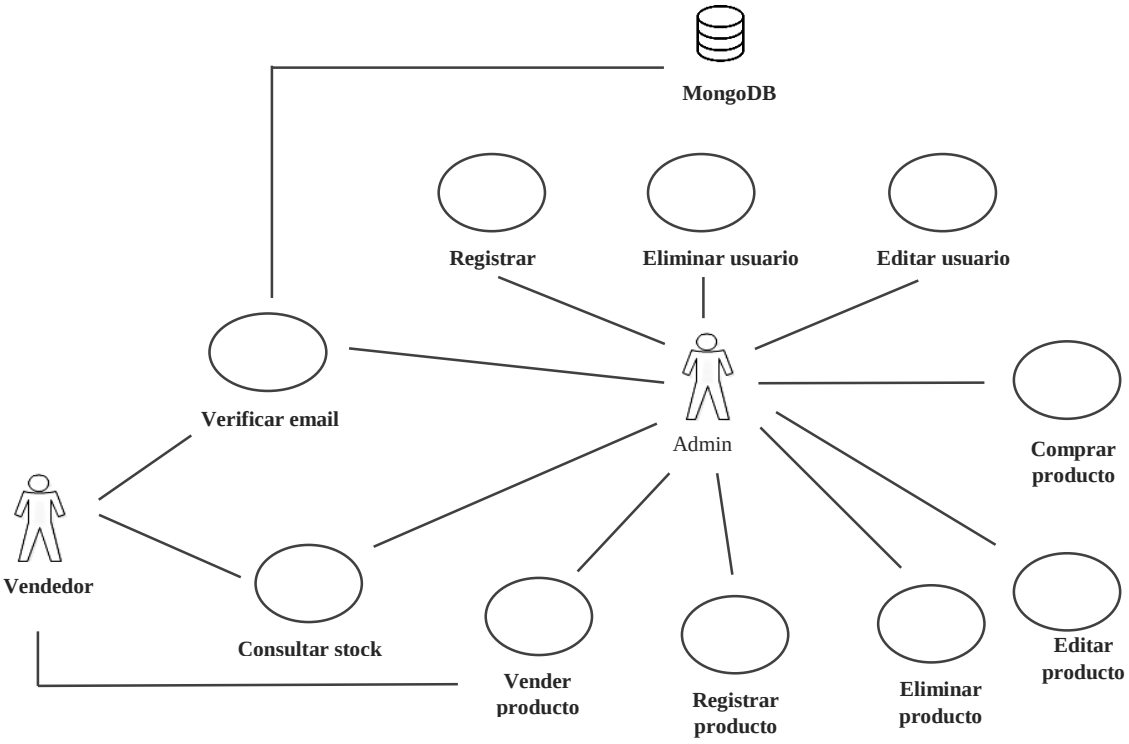

***Figura 8:** Diagrama de casos de uso Sistema Web*

***Elaborado por:** El autor, 2022*

Requerimientos funcionales

En la Tabla 5, se presenta los requerimientos funcionales del sistema web, estos fueron tomados directamente de la persona que lleva la contabilidad del almacén del taller.

Tabla 5: *Requerimientos funcionales del sistema web*

ID	REQUERIMIENTO	DESCRIPCION	PRIORIDAD
RF1	Acceso al sistema	El sistema web debe permitir el acceso a través de un correo electrónico y una contraseña. Este módulo debe identificar claramente el tipo de rol que se ha asignado al usuario que está por ingresar.	Alta
RF2	Registrar categorías	El sistema web debe permitir al administrador y vendedor ingresar una o más categorías de los productos a vender.	Alta
RF3	Registro de productos	El sistema web debe permitir al administrador y vendedor ingresar los productos que físicamente se encuentran en stock. Se debe crear un formulario que permita añadir imágenes de cada producto, cantidad, precio de compra y venta, descripción y, además, debe permitir agregar un puntaje a dicho producto.	Alta
RF4	Registro de clientes	El sistema web debe permitir al administrador y vendedor registrar los datos de un cliente.	Alta
RF5	Registro de usuarios	El sistema web debe permitir únicamente al administrador ingresar uno o más usuarios, a su vez puede designar el rol que crea conveniente para ese usuario, en este caso los roles son: 'ADMIN' y 'SELLER'	Alta
RF6	Registro de ventas	El sistema web debe permitir a los usuarios administrador y vendedor acceder al inventario de productos para poder efectuar una venta. Este corresponde el módulo principal del sistema web	Alta

Elaborado por: *el autor, 2022*

Requerimientos no funcionales

Estos requisitos representan atributos de calidad del sistema web, y corresponden a peticiones que el administrador del sistema exige en cuanto a funcionalidad y apariencia del sistema, la Tabla 6 muestra estos requisitos.

Tabla 6: *Requerimientos no funcionales del sistema web*

ID	REQUERIMIENTO	DESCRIPCION	PRIORIDAD
RNF1	Experiencia de usuario	El sistema web debe generar una percepción positiva por parte de los usuarios o administradores.	Alta
RNF2	Versatilidad	El sistema web debe ser ejecutable desde cualquier navegador	Alta
RNF3	Rendimiento	El sistema web será evaluado con la herramienta JMeter en base a su rendimiento	Alta

Elaborado por: *El autor, 2022*

3.5.2 Tablero Kanban Segunda Etapa

En esta etapa se presenta nuevas tareas asignadas a la columna Pendiente y En Proceso, de igual manera se ubican en la columna Finalizada las tareas que ya se cumplieron en la primera etapa, en la Tabla 7 se representan las actividades planificadas.

Tabla 7: *Actividades planificadas en la Segunda Etapa*

PENDIENTE	EN PROCESO	FINALIZADA
Realizar el diseño de la interfaz gráfica de usuario del sistema web con el software Pencil	Revisar la documentación de TypeScript	Realizar la toma de requerimientos funcionales y no funcionales.
Realizar el modelado de datos con el software Erwin Data Modeler	Revisar la documentación de Node y Express	Realizar el diseño del logo empresarial
Realizar la instalación de NodeJS, ExpressJS, AngularJS.	Revisar la documentación de MongoDB	Realizar el diagrama de casos de uso del sistema web
Instalar Postman y Robo3T	Revisar la documentación de Angular	

Elaborado por: *El autor, 2022*

3.5.2.1 Diseño de la interfaz

Para el diseño de la interfaz del sistema web se utilizó el software Pencil, esta herramienta es muy utilizada para el diseño de wireframes, se debe tomar en cuenta que esta es solo una

representación conceptual, el diseño del prototipo final puede estar sujeto a cambios, según las peticiones del administrador.

Login

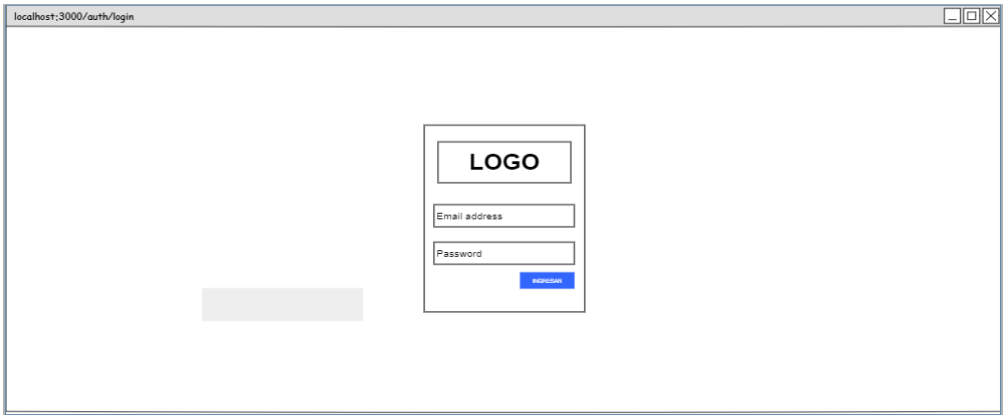

Figura 9: *Diseño del login*

Elaborado por: *El autor, 2022*

Vista principal del sistema

Figura 10: *Diseño del módulo principal - Dashboard*

Elaborado por: *El autor, 2022*

Vista del módulo de productos

Figura 11: *Diseño del módulo de productos*

Elaborado por: *El autor, 2022*

Vista del módulo de categorías

Figura 12: *Diseño del módulo de categorías*

Elaborado por: *El autor, 2022*

Vista del módulo de clientes

Figura 13: *Diseño del módulo de clientes*

Elaborado por: *El autor, 2022*

Vista del módulo de usuarios

Figura 14: *Diseño del módulo de usuarios*

Elaborado por: *El autor, 2022*

Vista del módulo de ventas

Figura 15: *Diseño del módulo de ventas*

Elaborado por: *El autor, 2022*

3.5.2.2 Diseño de la arquitectura

El sistema web diseñado utiliza una arquitectura tanto para el Frontend como para el Backend. A nivel de Frontend se ocupa el patrón MVVM, una variante del MVC; a nivel de Backend se utiliza el patrón Maestro-Detalle, con el cual se puede crear relaciones de tipo clave-valor.

Vista lógica. – MVVM, el controlador es reemplazado por la capa ViewModel la cual actúa de mediador entre la interfaz de usuario y la capa lógica.

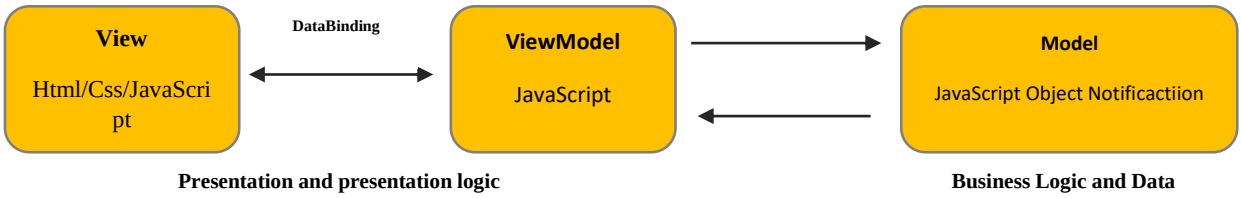

Figura 16: *Arquitectura Model ViewModel Model*

Elaborado por: *El autor, 2022*

Modelado de datos

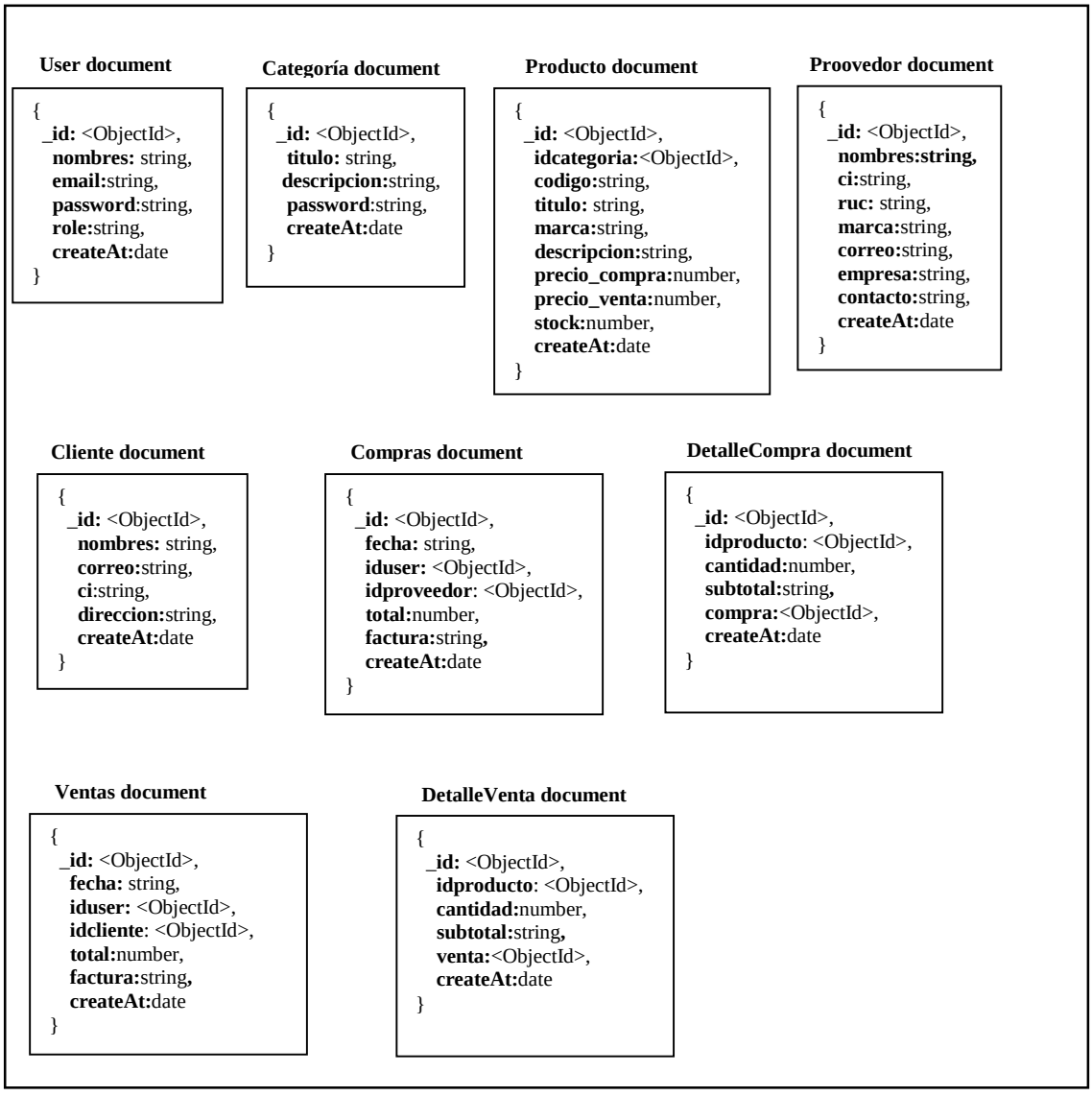

Figura 17: *Modelado de datos a implementar en MongoDB*

Elaborado por: *El autor, 2022*

La Figura 17 muestra el modelado de documentos para la elaboración de la base de datos en MongoDB. Con esta estructura se puede manipular datos de diferentes documentos al

insertar el identificador de un documento dentro de otro, este modelado se aparta de la normalización tradicional en SQL, pero a su vez presenta la ventaja de manipular y recuperar datos relacionados en una sola petición.

3.5.3 Tablero Kanban Tercera Etapa

En esta etapa de desarrollo se agregan nuevas tareas a la columna de Pendiente, estas corresponden al desarrollo Full Stack del sistema web con el framework Mean Stack, de igual manera se trasladan las tareas a la columna En Proceso y se ubican las tareas completadas a la columna Finalizada.

Tabla 8: *Actividades planificadas en la Tercera Etapa*

PENDIENTE	EN PROCESO	FINALIZADA
Desarrollo del Backend: creación del servidor, base de datos, modelos, controladores, servicios y rutas.	Realizar el diseño de la interfaz gráfica de usuario del sistema web con el software Pencil	Realizar la toma de requerimientos funcionales y no funcionales.
Pruebas de funcionamiento del Backend con Postman.	Realizar el modelado de datos con el software Erwin Data Modeler	Realizar el diseño del logo empresarial
Desarrollo del Frontend: componentes, modelos, servicios y rutas.	Realizar la instalación de NodeJS, ExpressJS, AngularJS.	Realizar el diagrama de casos de uso del sistema web
Pruebas de funcionamiento del Frontend con la herramienta de desarrolladores del navegador Chrome.	Instalar Postman y Robo 3T	Revisar la documentación de TypeScript
		Revisar la documentación de Node y Express
		Revisar la documentación de MongoDB
		Revisar la documentación de Angular
		Revisar la documentación de Node y Express

Elaborado por: *El autor, 2022*

Implementación del prototipo del sistema web

Para la implementación del sistema web del proyecto de investigación se requirió de las siguientes herramientas:

Tabla 9: *Herramientas para el desarrollo del sistema web*

HERRAMIENTA	DESCRIPCIÓN
IDE Visual Studio Code 64bits versión 1.6	Este es un potente IDE de desarrollo de software web, tiene plugins, snipets, extensiones y muchos recursos que agilizan el trabajo de desarrollo.
AngularJS versión 10.2.1	Angular/cli es la interfaz de líneas de comando utilizada para el desarrollo y escafolding del código que se utilizó para el frontend. Ideal para desarrollar aplicaciones web SPA.
NodeJs versión 12.14	Este es el entorno de ejecución de JavaScript para desarrollo del backend
NPM	Es el gestor de paquetes que necesita NodeJs para poder integrar Express, de igual manera lo utiliza Angular para descargar actualizaciones.
Express	Este constituye el marco de trabajo para crear el servidor que ejecutará el backend, también se lo utiliza para desarrollar las APIs.
Git Bash	Es el entorno de ejecución para Windows, con este se levanta la aplicación por el lado del frontend.
Robo3T	Es el gestor de bases de datos de MongoDB, con esta herramienta se puede manipular los datos de forma gráfica.
MongoDB	Es la base de datos NoSql que se utilizó para la creación de la base de datos del sistema web.

Elaborado por: *El autor, 2022*

3.5.3.1 Desarrollo del Backend

Para el desarrollo del Backend se empezó por crear una carpeta donde se alojará todos los recursos para la creación del servidor. Desde el terminal de Windows, se procede a ubicar la carpeta, una vez ahí se ejecuta el comando **npm init**, este comando crea un paquete de tipo JSON denominado **package.json,** este corresponde al principal en el proyecto ya que aquí se encuentran todas las dependencias de Node, fundamentales para que el Backend funcione correctamente, este proceso se muestra en la Figura 18.

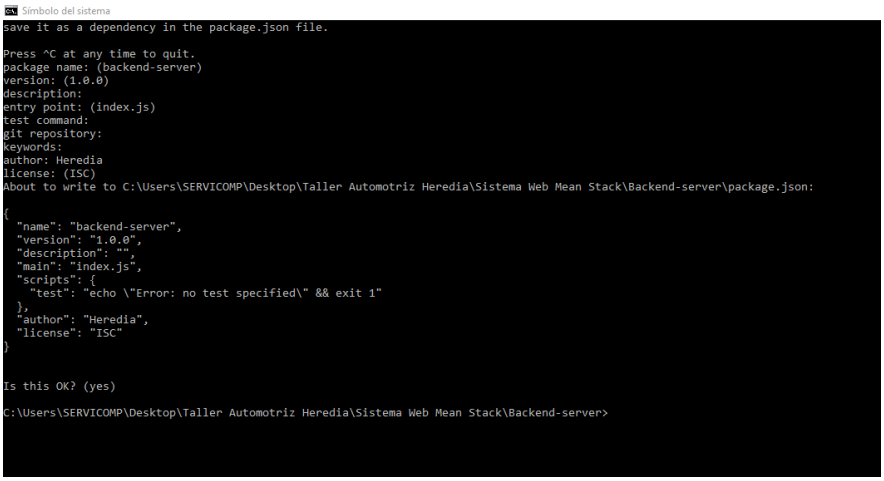

Figura 18: *Creación del entorno de desarrollo del Backend*

Elaborado por: *El autor, 2022*

Instalación de paquetes

A continuación, se procede a la instalación de los paquetes y dependencias de Node con el comando **npm i express bcrypt-nodejs body-parser connect-multiparty jsonwebtoken jwt-simple moment mongoose --save.** Estos paquetes tienes las siguientes funcionalidades:

- Express. – es el entorno de trabajo de NodeJS para la creación del servidor, también proporciona paquetes y librerías para el control de funciones del servidor web.
- Bcrypt-nodejs. – este módulo permite encriptar las contraseñas que se ingresarán a través del login.
- Body-parser. – este módulo de middleware permite manejar las solicitudes de los métodos HTTP, en especial POST, parte de analizar las peticiones de tipo JSON.
- Jsonwebtoken. – con esta librería se generan los tokens los cuales son indispensables para la creación de roles y seguridad en el Backend
- Mongoose. – esta es una librería fundamental para la conexión de MongoDB con el servidor, su ejecución se produce en tiempo real.

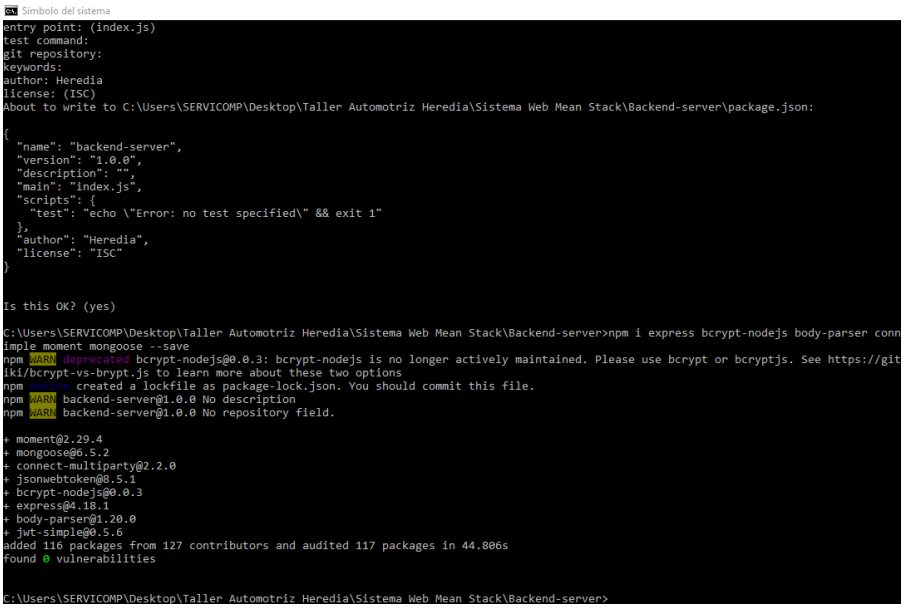

Figura 19: *Instalación de paquetes y dependencias de Node*

Elaborado por: *El autor, 2022*

Conexión entre MongoDB y Robo3T

Una vez que las dependencias, módulos y librerías de Node están instaladas, se procede a ejecutar MongoDB como demonio, seguido a eso se realiza la conexión con el gestor Robo3T a través del puerto 27017. En la Figura 20, se muestra la conexión con Robo3T a través del puerto 27017.

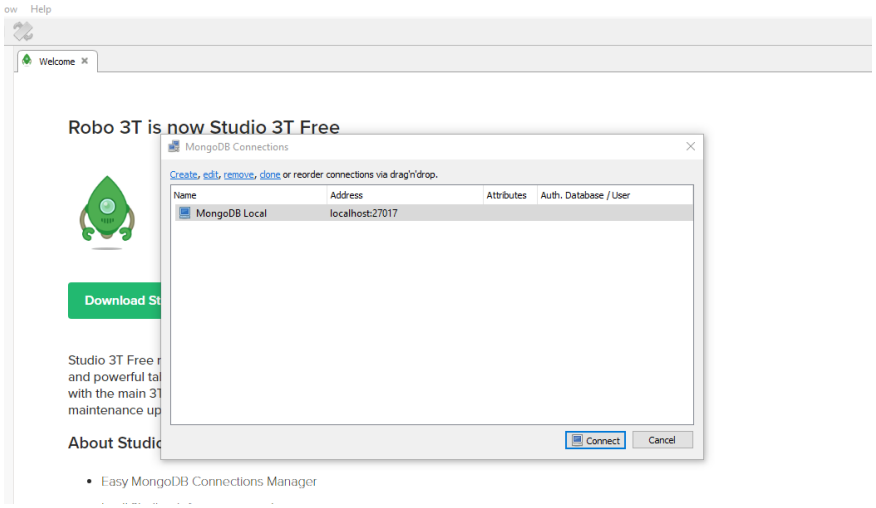

Figura 20: *Conexión con MongoDB y Robo3T*

Elaborado por: *El autor, 2022*

Creación del servidor web

Se implementó el siguiente código para crear la conexión del servidor con la base de datos en MongoDB, en la Figura 21 se muestra el código en JavaScript el cual permite que el

servidor y la base de datos interactúen entre sí, el archivo encargado de realizar esta petición se lo denominó como **app.js.**

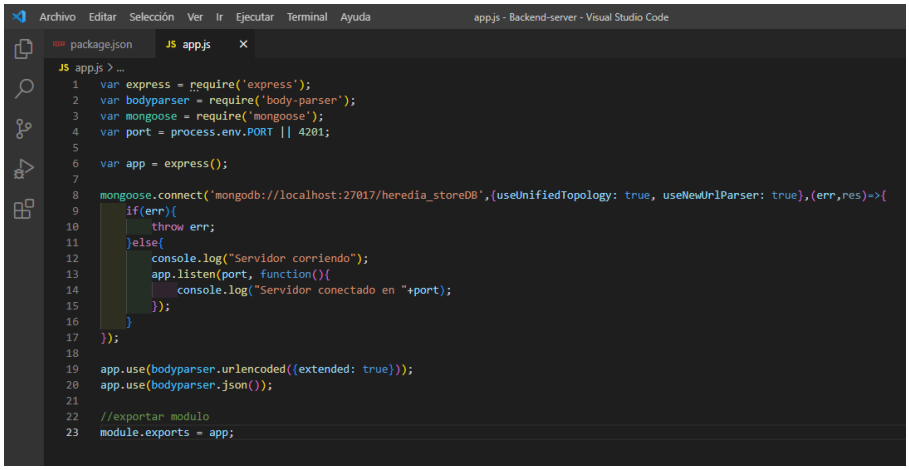

Figura 21: *Conexión del servidor con la base de datos*

Elaborado por: *El autor, 2022*

Creación de los modelos de la base de datos

La creación de los modelos de la base de datos corresponde al desarrollo del servidor con NodeJS y Express, estos modelos contienen las variables para el almacenamiento de datos dentro de una colección de documentos en MongoDB, cada documento contiene un esquema de datos con diferentes tipos de variables. En la Figura 22, se muestra el modelo User con un esquema estructurado para el almacenamiento de datos de un usuario, de esta manera se representan los demás modelos de la base de datos como son: categoría, cliente, productos, compras, ventas, proveedores y usuarios.

Figura 22: *Estructura del modelo User*

Elaborado por: *El autor, 2022*

Controladores de los modelos

En cada controlador se desarrollaron funciones de inserción, eliminación, edición y listado de datos de cada uno de los modelos antes creados. En la Figura 23, se muestra el controlador de la entidad Venta, el cual contiene tres variables requeridas para cumplir con las reglas de venta. Todas las funciones de los controladores reciben dos parámetros de tipo Request y Response, esto permite tener un mejor control de errores de tipo 500, 402 o 403, los más comunes al momento de desarrollar un servidor. Finalmente, estas funciones son exportadas al módulo de rutas, en el prototipo se lo denomino como **routes**.

Controlador de la Entidad Venta

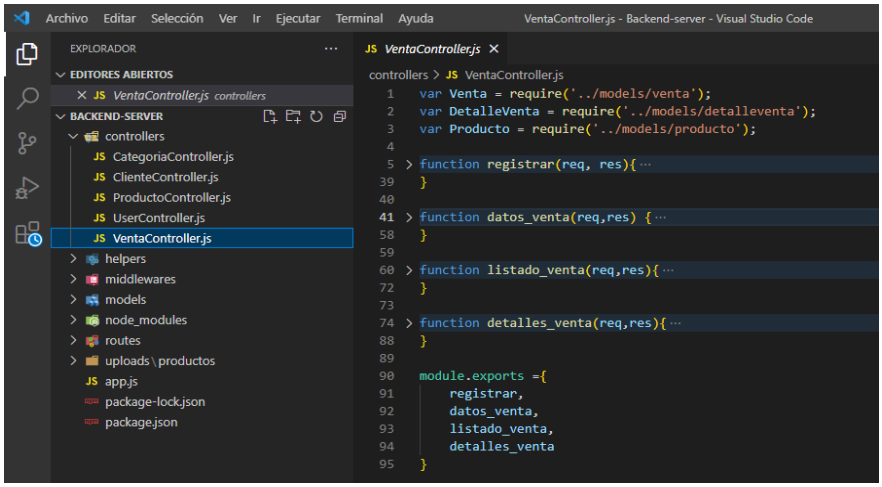

Figura 23: *Controlador del modelo Ventas.*

Elaborado por: *El autor, 2022*

Creación de las rutas para los modelos

En este módulo se crearon las rutas para realizar las peticiones al servidor a través de los distintos métodos de HTTP, en la Figura 24, se presenta las rutas de la entidad Producto, aquí se utilizó una variable de express y como en los casos de los modelos estas rutas se deben exportar para su comunicación. La metodología es la misma para los demás modelos y controladores, se cita un método de HTTP, se instancia el controlador de la entidad y finalmente se hace un llamado a la función que se planea utilizar.

Ruta para la Entidad Producto

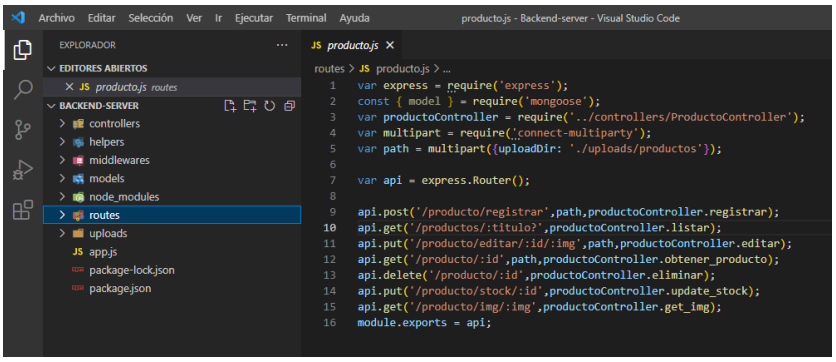

Figura 24: *Ruta para la entidad Producto*

Elaborado por: *El autor, 2022*

La Figura 25, muestra el módulo **app.js**, finalizado, con las rutas implementadas, este módulo es independiente y controla cada ruta creada para cada modelo y controlador.

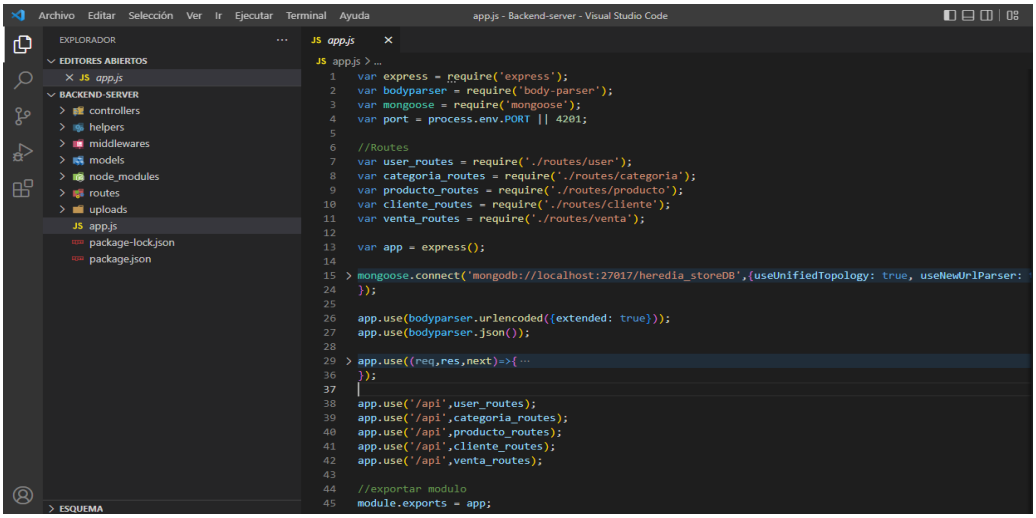

Figura 25: *Modulo app con las rutas del servidor*

Elaborado por: *El autor, 2022*

Pruebas de funcionamiento del Backend con Postman

Como se explicó anteriormente, Postman es una herramienta que permite realizar una prueba de funcionalidad al servidor a través de métodos HTTP. En la Figura 26, se aprecia la creación de una categoría a través del método post con la ruta: **http://127.0.0.1:4201/api/categoria/registrar**. Las pruebas de funcionamiento del Backend con Postman se realizaron para cada una de las entidades del servidor.

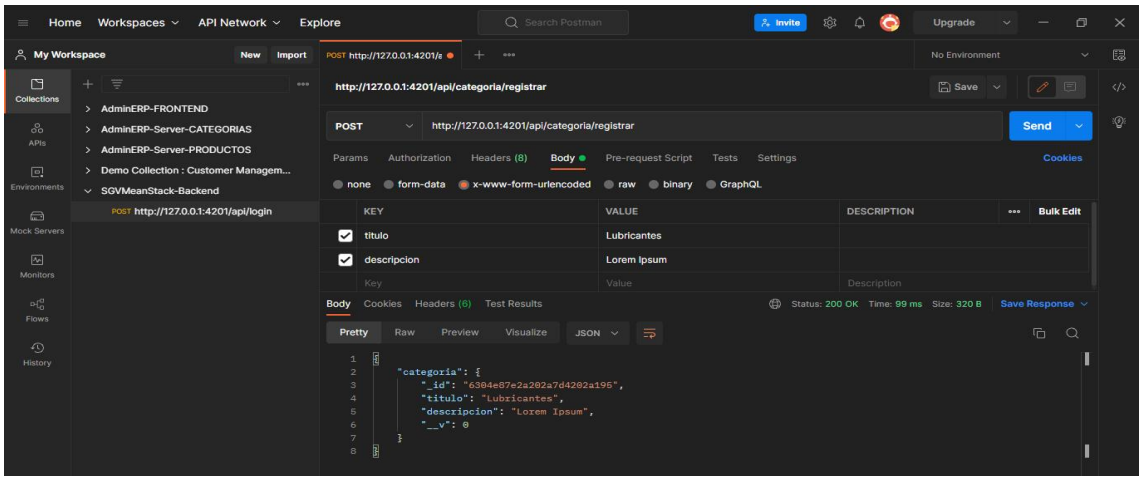

Figura 26: *Insertar categoría desde Postman*

Elaborado por: *El autor, 2022*

3.5.3.2 Desarrollo del Frontend

Para el desarrollo de la interfaz gráfica se utilizó el framework AngularJS en su versión 10.12 con el paquete de NodeJS en su versión 12.14. En la Figura 27, se aprecia la instalación de los paquetes de AngularJS desde el terminal de Windows.

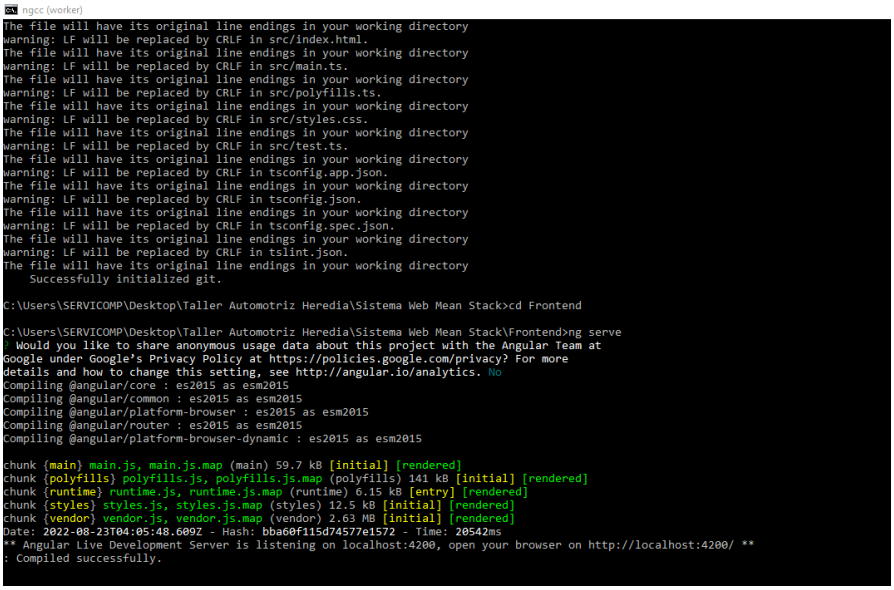

Figura 27: *Instalación de AngularJS y sus módulos*

Elaborado por: *El autor, 2022*

La interfaz gráfica de usuario corresponde al principal desarrollo del sistema web, a través de esta el usuario 'administrador' o 'vendedor' interactúa con la base de datos desarrollada con MongoDB, Express y NodeJS. En el desarrollo de la interfaz se tomó en cuenta el

modelo-vista-modelo de la vista (MVVM), patrón arquitectónico que utiliza Angular para el desarrollo de aplicaciones web SPA.

Se subdividió la interfaz en 8 carpetas principales para el funcionamiento del sistema, las cuales son:

- **Auth,** para el control de acceso de usuarios la cual contiene el módulo de login.
- **Guards,** para la seguridad de las rutas de acceso a los componentes, de esta manera se previene que no se pueda ingresar a ningún componente del sistema sin antes haber pasado por los filtros de seguridad implementados en el Backend y Frontend.
- **Interfaces,** en esta carpeta se alojan los archivos TypeScript necesarios para el control de formularios de ingreso de datos al sistema tales como: formularios de registro y edición de clientes, usuarios, productos, categorías, proveedores, compra y venta.
- **Models,** de igual manera que en el Backend el desarrollo del Frontend con Angular necesita manipular la lógica de datos a través de las clases creadas para cada entidad, de esta manera se cumple el patrón arquitectónico mvvm, los modelos se manipulan desde la vista.
- **Pages,** en esta carpeta se ubican todos los componentes que conforman el sistema: usuarios, clientes, proveedores, productos, compras y ventas.
- **Services,** aquí se encuentran los servicios con los métodos HTTP, Headers y Cors, necesarios para la comunicación con el Backend, cada vez que se requiera realizar una petición al servidor se debe pasar por este filtro de acceso.
- **Shared,** esta carpeta contiene la estructura de la vista del sistema web. Angular es un framework que permite realizar sistemas o aplicaciones web que no recargan la página, una buena práctica de programación que recomienda Angular es subdividir la estructura de la página de manera que lo único que cambie sean los módulos internos, para mejor entendimiento se puede considerar como ejemplo Gmail, siempre se ve un menú lateral mientras se navega en los correos.
- Por último, pero no menos importante, se encuentra la carpeta **No Page Found** con las vistas por defecto, cuando no se encuentre una página solicitada aparece un mensaje de error o de redirección. En la Figura 28, se observa las carpetas mencionadas dentro del proyecto en Angular explicado en el paso anterior.

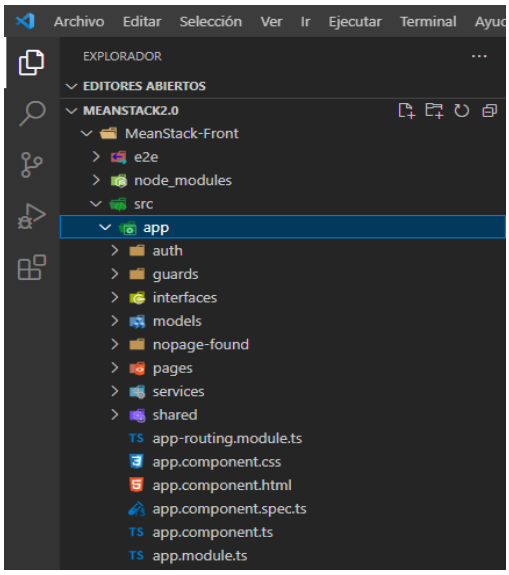

Figura 28: *División del proyecto en Angular*

Elaborado por: *El autor, 2022*

Desarrollo de la vista principal del Sistema Web

Para el diseño de la vista principal del sistema web se generaron cuatro componentes dentro de la carpeta Shared: **breadcrumbs, footer, header, sidebar**, de igual manera y, para cumplir con el propósito de crear una aplicación web SPA se procedió a crear un módulo independiente denominado **shared.module.ts** para agrupar todos estos componentes y exportarlos al módulo principal que por defecto Angular proporciona, este se denomina **app.module.ts.** En la Figura 29, se muestra los componentes y el módulo dentro de la carpeta Shared.

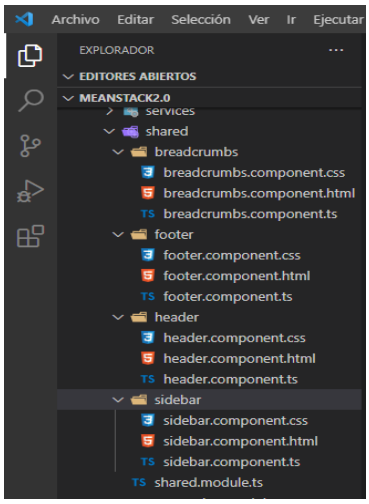

Figura 29: *Componentes de la carpeta Shared*

Elaborado por: *El autor, 2022*

La Figura 30, muestra la vista principal del sistema web desde el navegador.

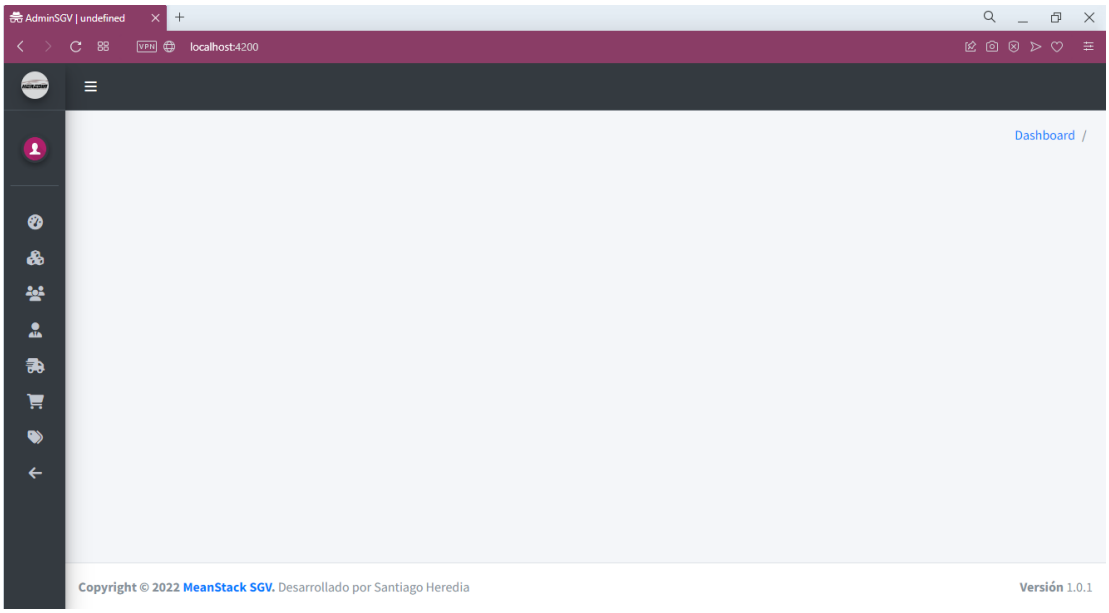

Figura 30: *Vista principal del sistema web*

Elaborado por: *El autor, 2022*

Desarrollo de los componentes de la carpeta Auth

Dentro de la carpeta Auth se procedió a generar un componente denominado Login, a través del cual se realiza el filtrado de acceso a usuarios según su correo electrónico con el rol 'administrador' o 'vendedor'. De igual manera se generó dos módulos de acceso independiente denominados: **auth.module.ts,** y **auth-routing.module.ts**, de esta manera todos los componentes que se crearan en la carpeta Auth tienen un control de rutas y componentes independiente al resto del sistema web.

Maquetación del componente Login

En la maquetación del componente Login se desarrolló un formulario con 3 elementos <input> para capturar el correo, contraseña y un valor booleano denominado guardar sesión, este último almacena en el Local Storage del navegador el correo electrónico del usuario que ingresa de tal manera que pueda evitar escribir nuevamente el correo electrónico. La Figura 31, muestra el formulario creado con etiquetas HTML.

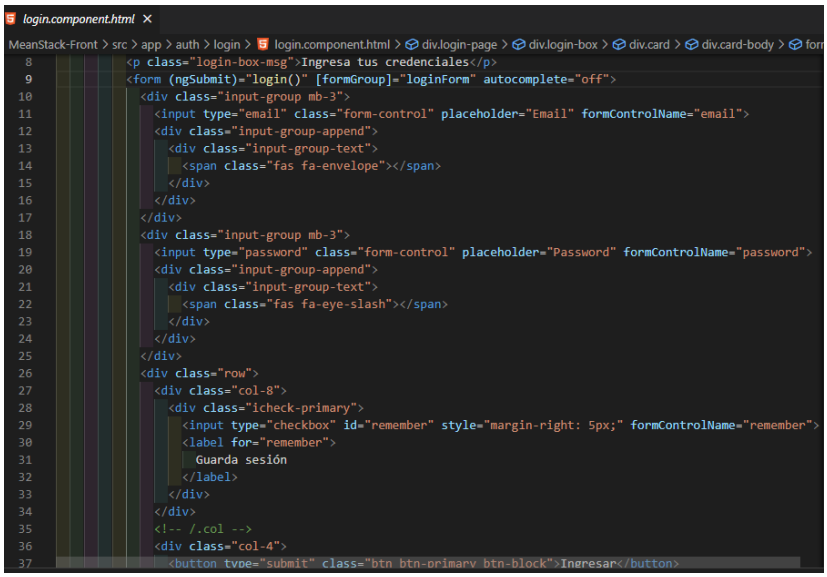

Figura 31: *Maquetación con HTML del componente Login*

Elaborado por: *El autor, 2022*

La lógica de programación se desarrolló utilizando TypeScript, en esta sección se controla que tipo de usuario ingresa al sistema, y si antes de ingresar dio clic en guardar sesión en el almacenamiento local del navegador, el código se muestra en la Figura 32.

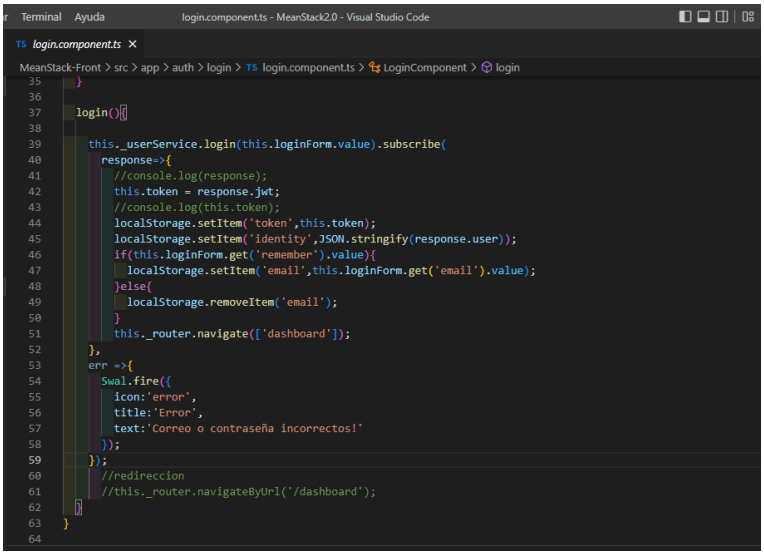

Figura 32: *Lógica de programación del componente Login*

Elaborado por: *El autor, 2022*

Servicios del componente Login

Dentro de la carpeta Services se generó un servicio para comunicación con el Backend, en este servicio también se crea un servicio que obtenga el token que genera el servidor a los usuarios que ingresan al sistema, de igual manera se crea un servicio que obtiene la identidad del usuario. En la Figura 33, se visualiza los servicios de este componente.

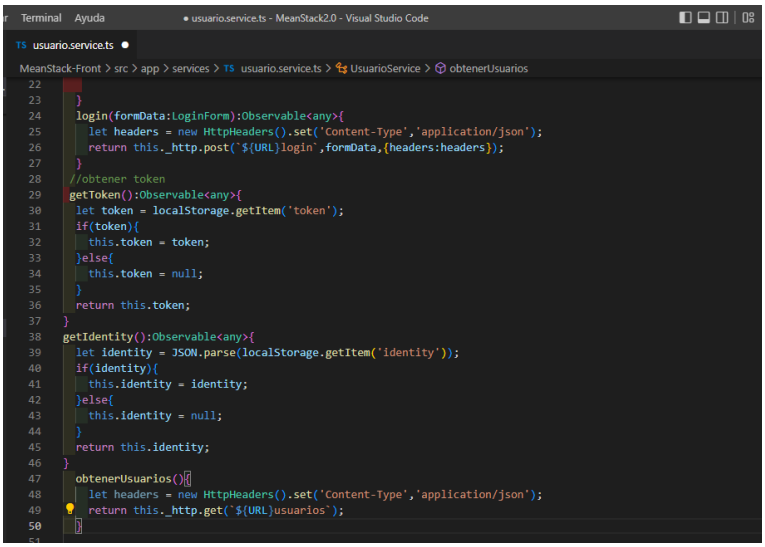

Figura 33: *Servicios del componente Login*

Elaborado por: *El autor, 2022*

Finalmente, se obtiene la vista principal del componente creado con sus servicios y rutas gestionadas. En la figura 34, se aprecia el resultado del proceso anterior.

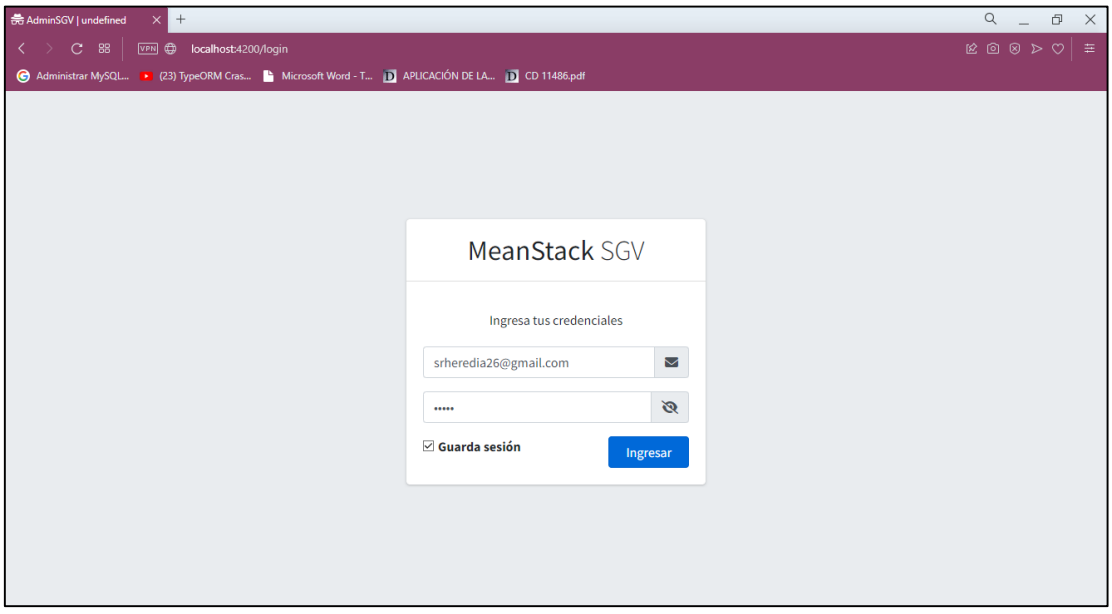

Figura 34: *Vista principal del componente Login*

Elaborado por: *El autor, 2022*

Desarrollo de los componentes de la carpeta Pages

Dentro de esta carpeta se encuentra los componentes responsables del desarrollo de la gestión de ventas del sistema web, estos son: Dashboard, Usuarios, Clientes, Proveedores, Categorías, Productos, Compras, Ventas, Reporte de ventas, Reporte de compras. De igual manera que en la carpeta Auth, también se creó un control independiente de módulos y rutas denominados: **pages.module,ts,** y **pages-routing.module.ts,** respectivamente.

Desarrollo del componente Dashboard

Siguiendo como base el wireframe descrito en el desarrollo de la interfaz con Pencil, se procedió con la maquetación del componente del tablero principal del sistema, aquí se muestra cuatro tarjetas informativas: total de compras, ventas, clientes y stock, también se añadió una tabla que muestra el estado de los artículos del almacén, si un artículo tiene 0 existencias este tablero los mostrara una pequeña alerta. En la Figura 35, se aprecia la maquetación con etiquetas HTML, esta sección no recibe parámetros de entrada por el usuario, en su lugar ocupa datos obtenidos de la base de datos.

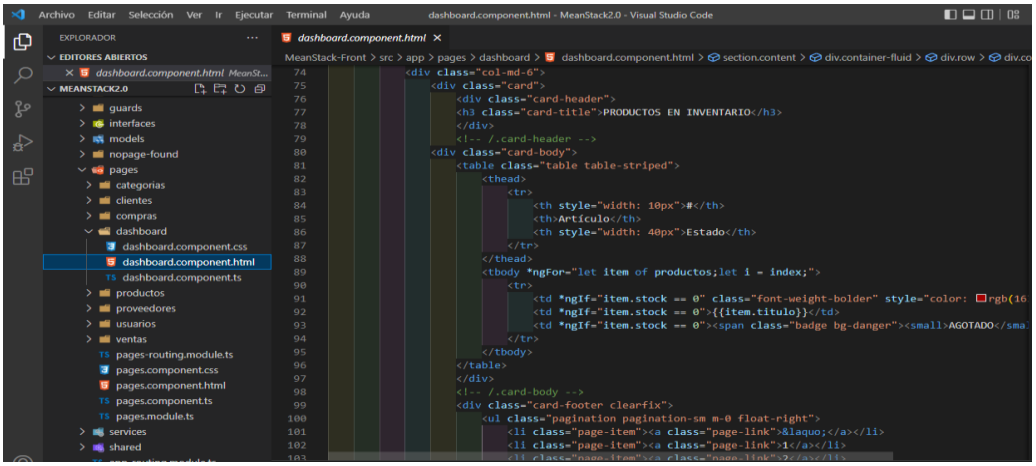

Figura 35: *Maquetación del Dashboard*

Elaborado por: *El autor, 2022*

Se procede a realizar la lógica de programación para obtener todos los datos ya almacenados en la base de datos, se divide los procesos en funciones que realizan peticiones a los diferentes servicios del sistema. Se puede apreciar esta implementación en la Figura 36.

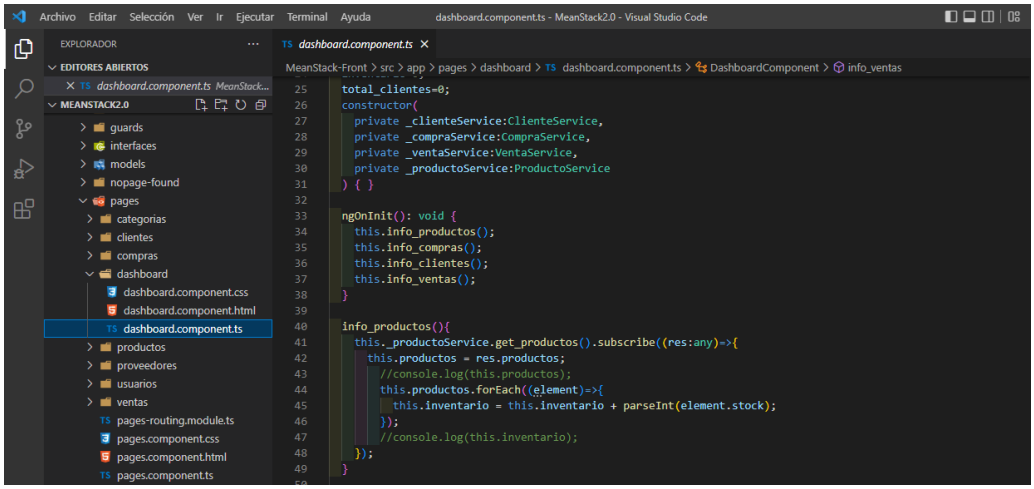

Figura 36: *Lógica de programación del componente Dashboard*

Elaborado por: *El autor, 2022*

Finalmente, se obtiene una vista principal del Dashboard en el navegador, véase la Figura 37.

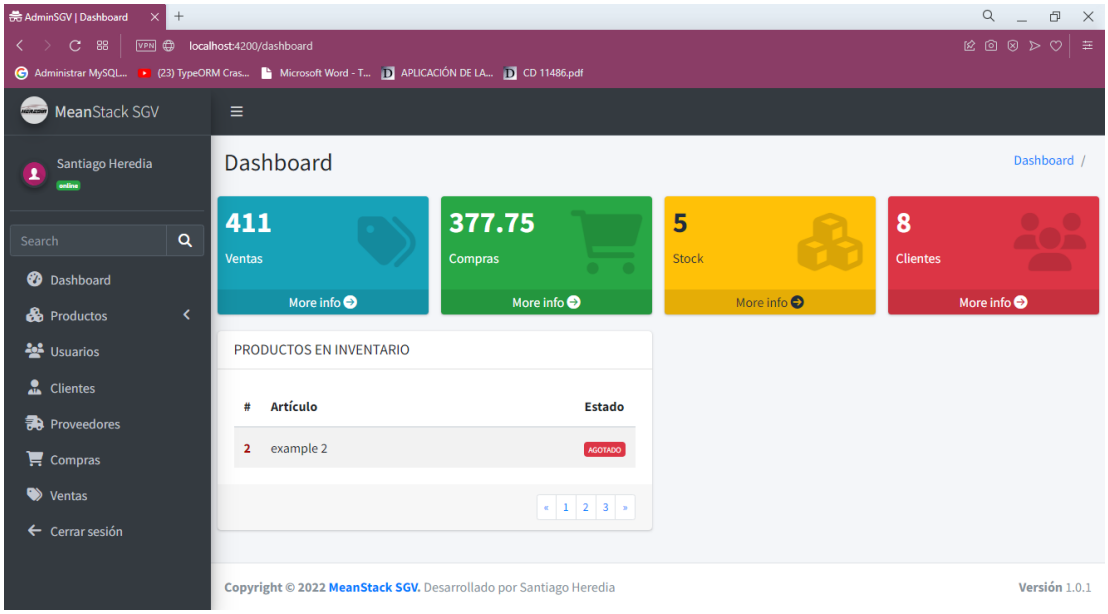

Figura 37: *Vista principal del componente Dashboard*

Elaborado por: *El autor, 2022*

Desarrollo del componente Usuarios

En esta sección se realizó el desarrollo del componente Usuarios el cual permite agregar, editar, consultar y eliminar usuarios con acceso al sistema web. Aquí se designa el tipo de rol que tendrá el usuario y, dependiendo de este, el usuario podrá acceder a todos o algunos de los componentes del sistema. En el formulario de registro el servidor valida que los correos electrónicos no se repitan, de esta manera el sistema no admite correos repetidos. En la Figura 38, se muestra la maquetación del componente, en esta se incluye un modal para ingresar usuarios, botones de edición y eliminación, y una tabla con opción a filtrado de datos para tener una búsqueda más eficiente.

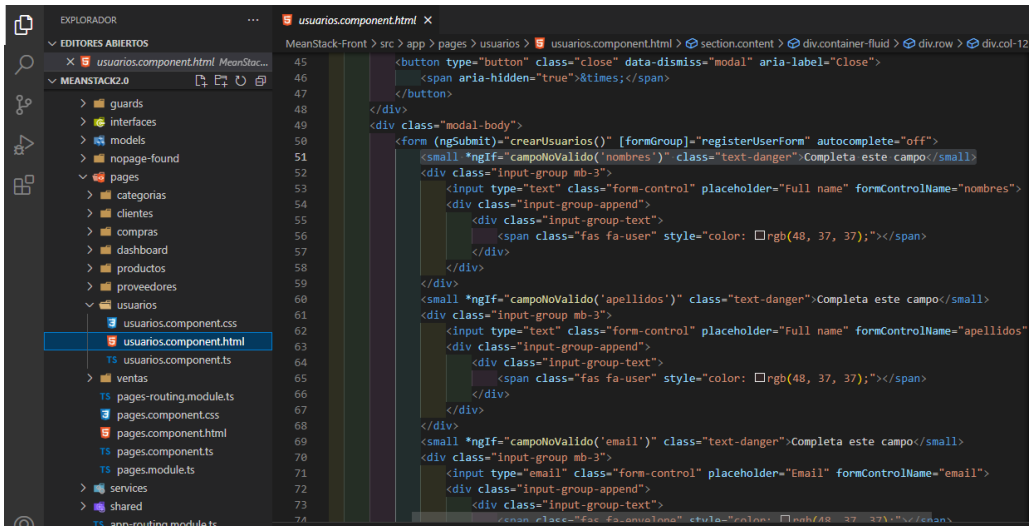

Figura 38: *Maquetación del componente Usuarios*

Elaborado por: *El autor, 2022*

En el desarrollo de la lógica de programación de los métodos: eliminar, editar, consultar y añadir de este componente, se tomó también en cuenta una función que previene la autoeliminación, en el caso de un usuario 'administrador' tendrá acceso a este componente y puede eliminar o editar a cualquier usuario, sin embargo, se debe prevenir que el usuario por accidente o curiosidad se auto elimine. La Figura 39, muestra la implementación del código.

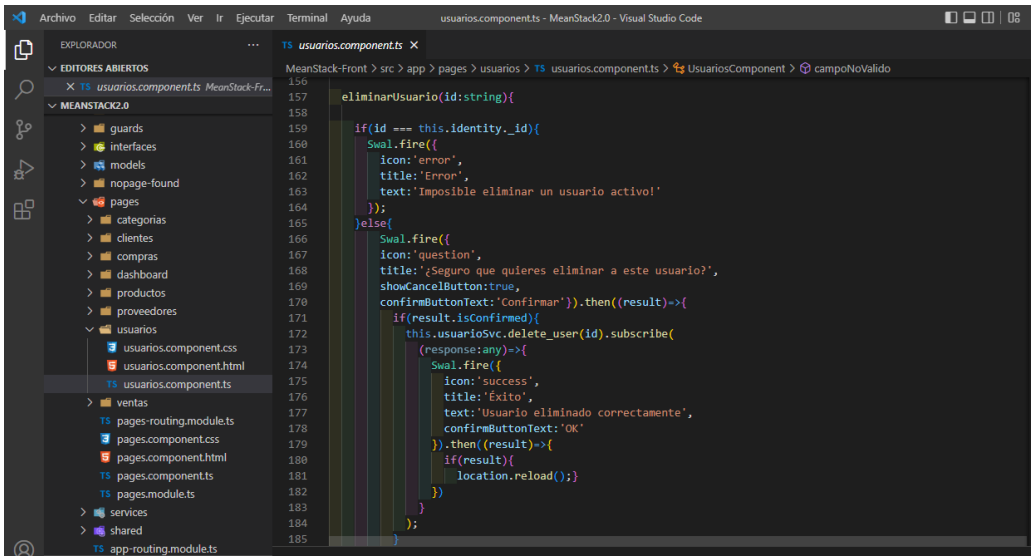

Figura 39: *Lógica de programación del componente Usuarios*

Elaborado por: *El autor, 2022*

La Figura 40, muestra la vista principal del componente de usuarios.

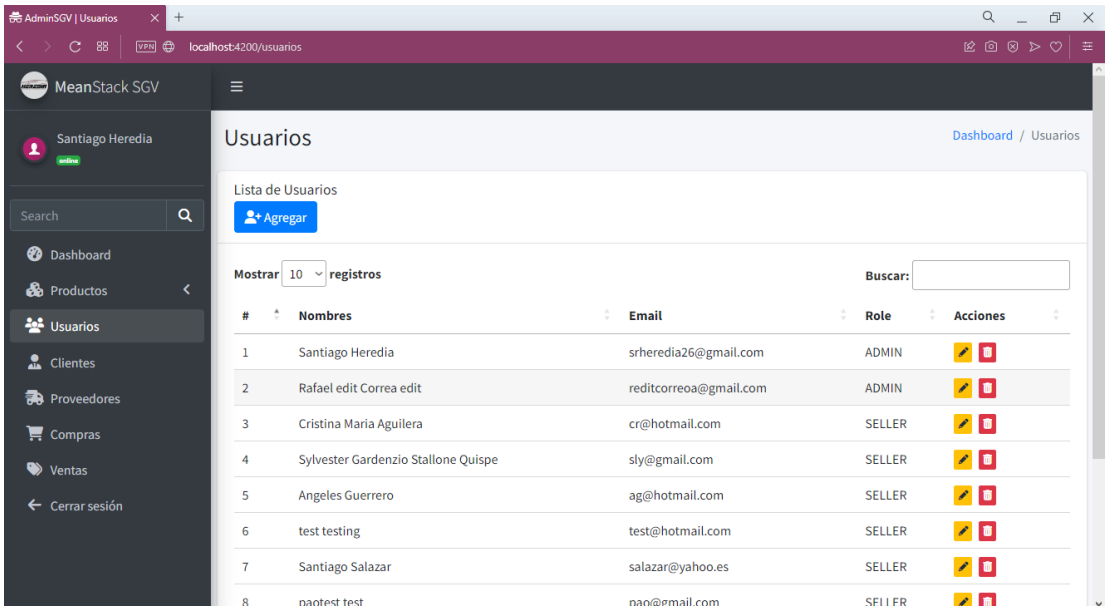

Figura 40: *Vista principal del componente Usuarios.*

Elaborado por: *El autor, 2022*

Desarrollo del componente Clientes

En este componente se registran los clientes que realizan compras al almacén, a través de un formulario se captan los datos relevantes como: cédula, nombres, email y contactos, al igual que el desarrollo de la lógica del componente Usuarios, el formulario de registro de clientes no admite ingresar una cédula incorrecta gracias a un algoritmo de validación de cédulas, de igual manera este dato no puede registrarse dos veces. En la Figura 41, se muestra la maquetación del componente con HTML.

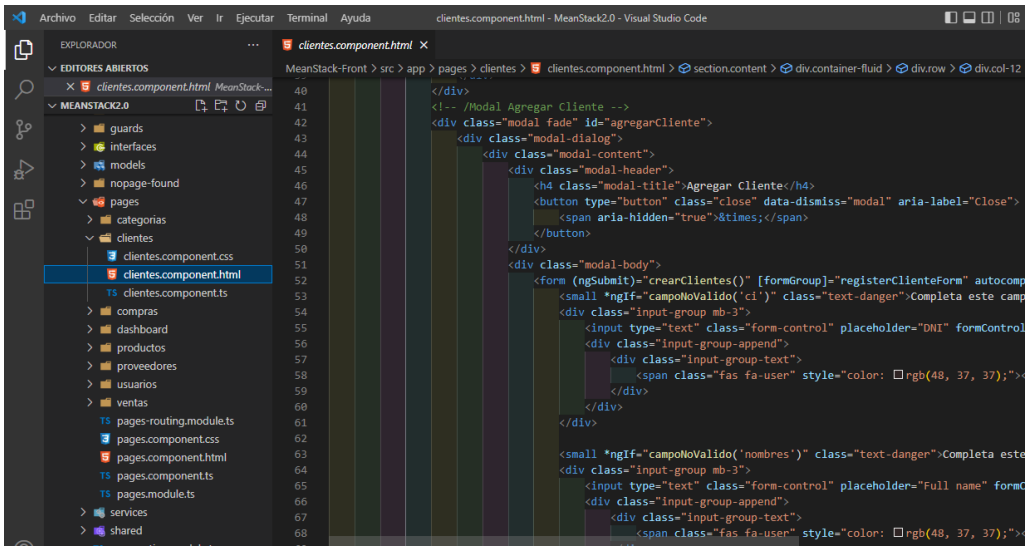

Figura 41: *Maquetación del componente Clientes*

Elaborado por: *El autor, 2022*

En cuanto a la lógica de programación del componente, se implementaron los métodos para agregar, eliminar, consultar y editar un cliente. La Figura 42, muestra estas funciones.

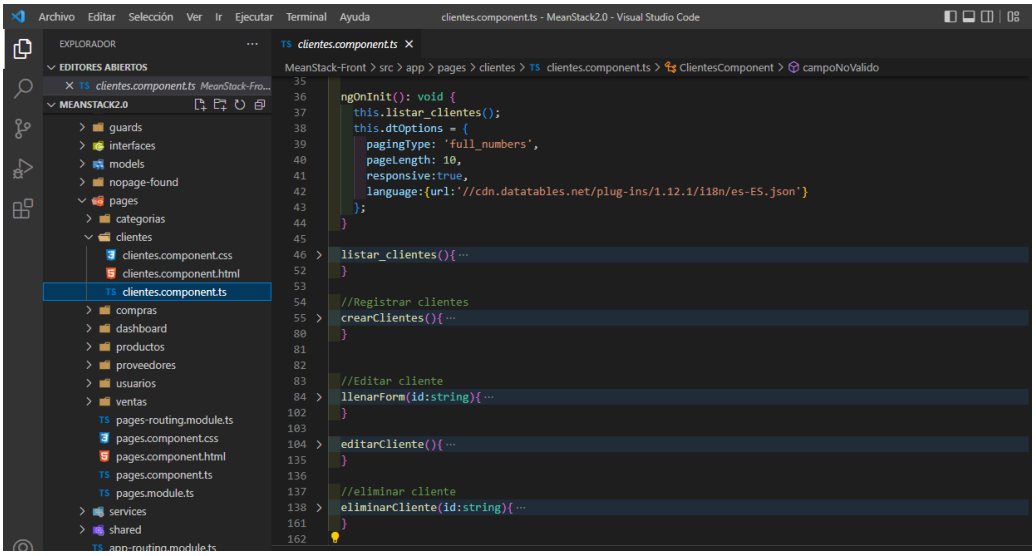

Figura 42: *Lógica de programación del componente Clientes*

Elaborado por: *El autor, 2022*

Finalmente, se obtiene una vista principal en el navegador como muestra la Figura 43.

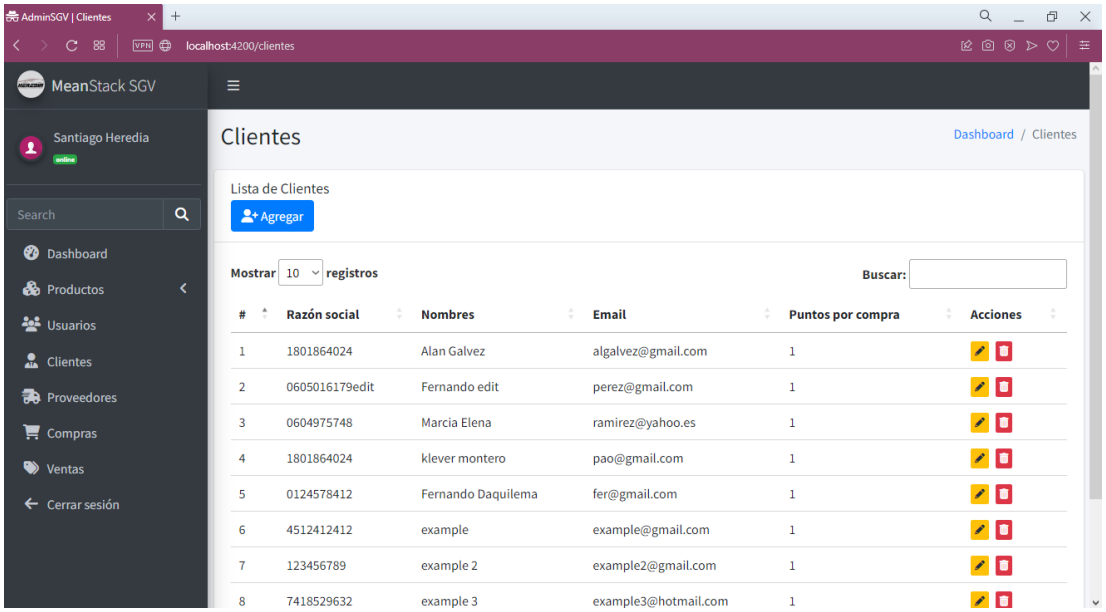

Figura 43: *Vista principal del componente Clientes.*

Elaborado por: *El autor, 2022*

Desarrollo del componente Proveedores

En este componente, se ingresa a través de un formulario los datos de los proveedores principales del almacén, este componente es importante ya que en el reporte de compras se detalla la persona y la empresa a la que representa un proveedor. Al igual que el

componente Clientes la cedula de identificación del proveedor pasa por un filtro de reconocimiento, no se puede continuar con el proceso de registro si la cedula es incorrecta o está registrada en el sistema. En la Figura 44, se muestra la maquetación del componente Proveedores.

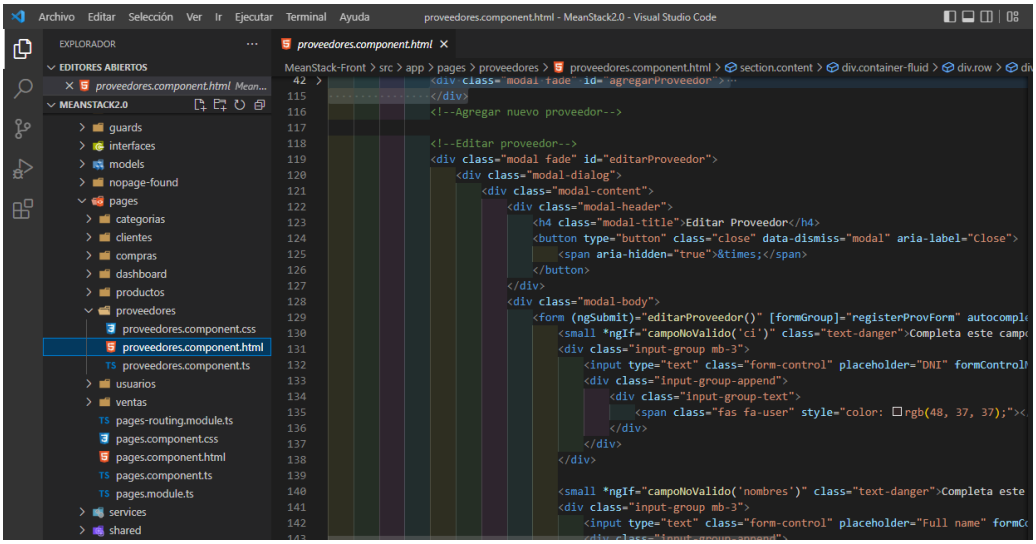

Figura 44: *Maquetación del componente Proveedores.*

Elaborado por: *El autor, 2022*

Para la lógica de programación se implementó los métodos de: agregar, eliminar, editar y consultar datos de un proveedor, como lo muestra la Figura 45.

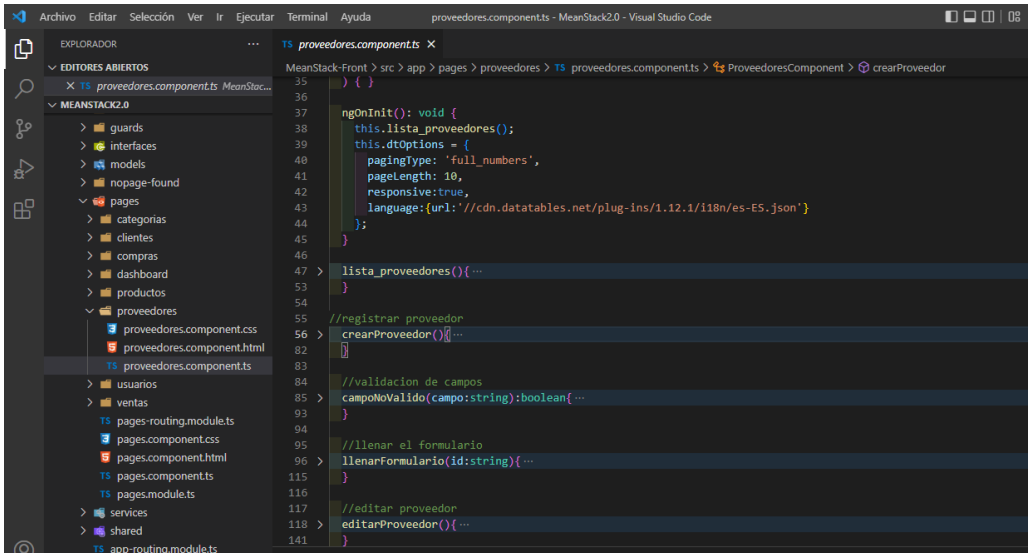

Figura 45: *Lógica de programación del componente Proveedores.*

Elaborado por: *El autor, 2022*

Finalmente, en la Figura 46, se muestra la vista principal del componente Proveedores desde el navegador.

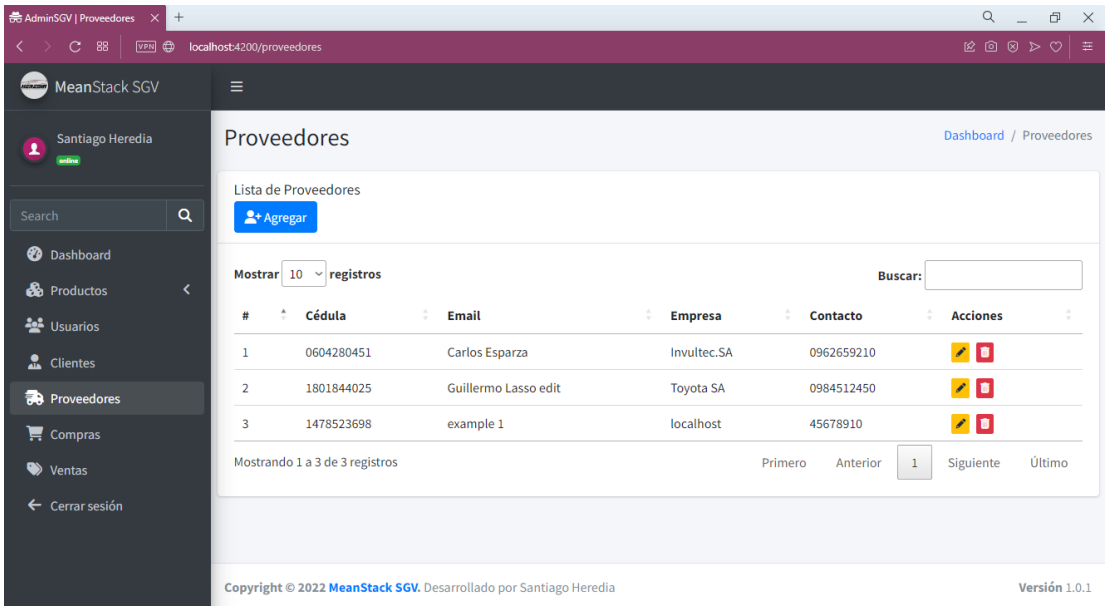

Figura 46: *Vista principal del componente Proveedores.*

Elaborado por: *El autor, 2022*

Desarrollo del componente Productos

En el menú lateral del sistema se muestra un item de nominado **Productos** el cual contiene dos submenús: Productos con el registro de artículos dentro del inventario; y Categorías, el cual agrupa todos los artículos ingresados categóricamente. A partir de este componente se emplea el patrón Maestro-Detalle que se implementó en el desarrollo del Backend, la función principal es traer todos los datos de una entidad a través de un objeto de tipo JSON, el cual se obtiene a través de una petición al servidor. En la Figura 47, se muestra como los Pipes llaman a la entidad Categorías para poder registrar un producto.

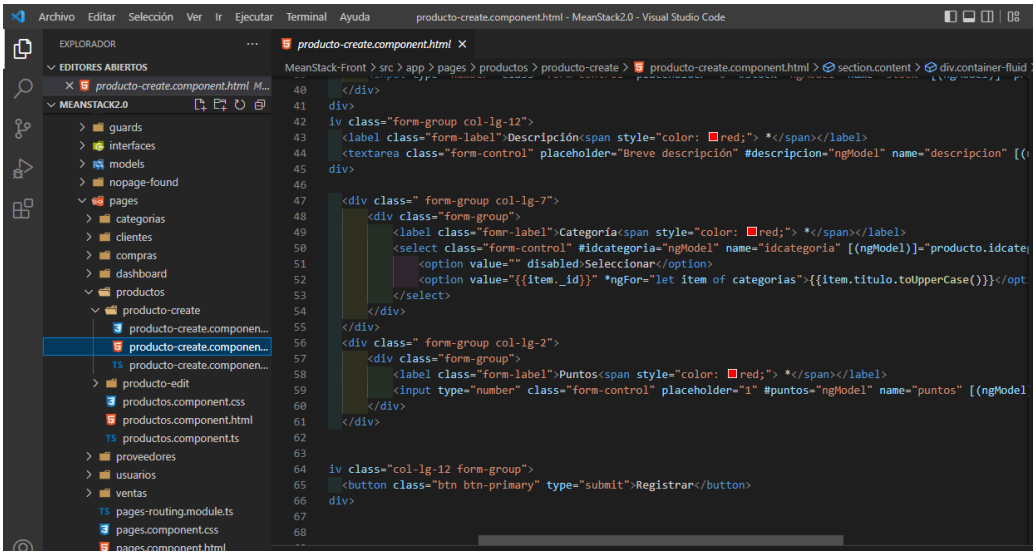

Figura 47: *Maquetación del componente Productos.*

Elaborado por: *El autor, 2022*

En la Figura 48, se muestra el código implementado en el componente crear producto.

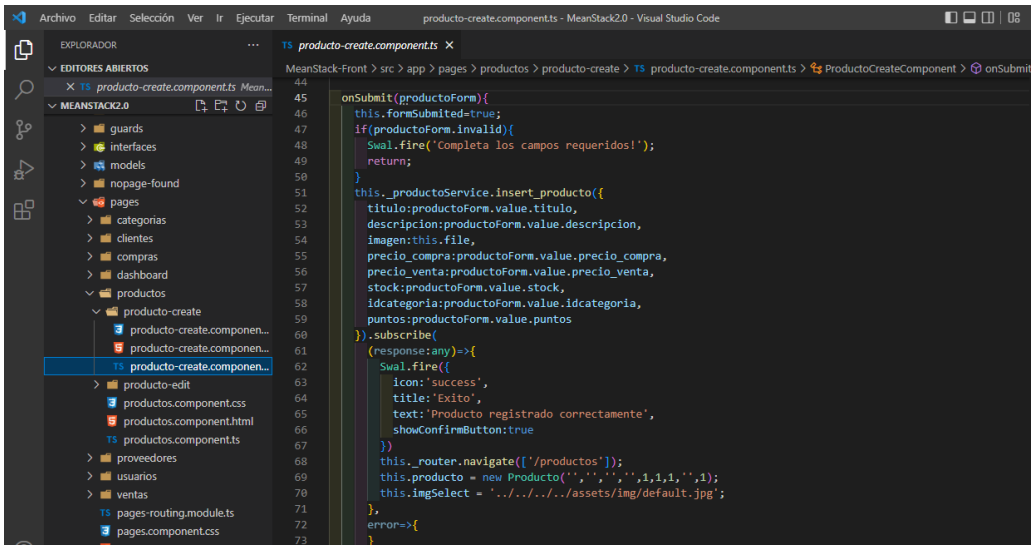

Figura 48: *Lógica de programación del componente Productos*

Elaborado por: *El autor, 2022*

Para el proceso de edición se reutilizó el mismo código de inserción de productos, con la diferencia que en el servicio el método **post** se sustituye por el método **put.** Finalmente, en la Figura 49, se aprecia la vista principal del componente Productos, al igual que en las demás vistas, esta tiene la opción de filtrar los datos las búsquedas individuales de objetos en la base de datos.

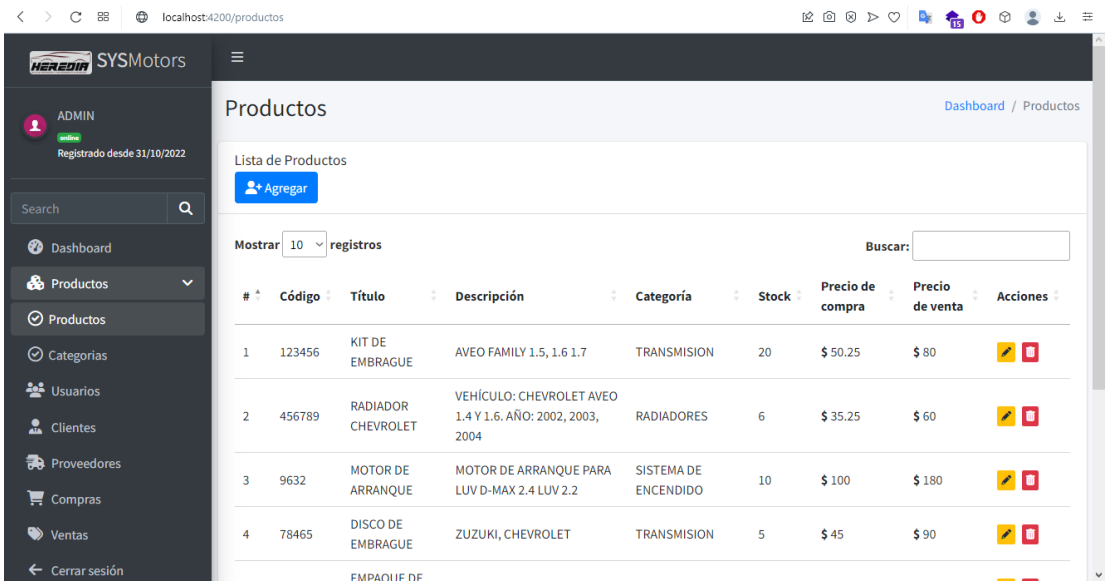

Figura 49: *Vista principal del componente Productos.*

Elaborado por: *El autor, 2022*

Desarrollo del componente Compras

Al igual que el componente Productos, el componente Compras utiliza el patrón Maestro-Detalle para acceder a los objetos Productos, Proveedores y Usuarios, la vista principal del componente de compras permite obtener un reporte individual de una compra con los datos específicos de la misma, los campos fecha y usuario son solo de lectura ya que estos se obtienen independientemente. En la Figura 50, se observa la maquetación del componente Compras.

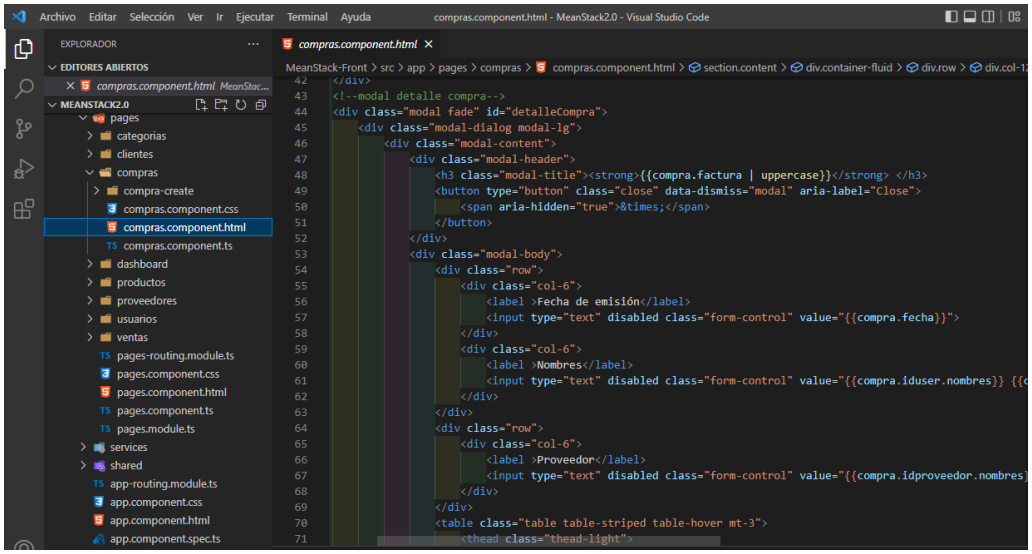

Figura 50: *Maquetación del componente Compras*

Elaborado por: *El autor, 2022*

En la lógica de programación se ubican las funciones y métodos para ingresar y listar los detalles de una compra, con la excepción que no se puede eliminar ninguna compra. Según los requerimientos del sistema web, se desea tener una gestión del control de ventas, sin embargo, se añadió el módulo de compras para conocer un estimado de la ganancia neta del almacén. La Figura 51, muestra las funciones que permiten registrar y consultar una compra.

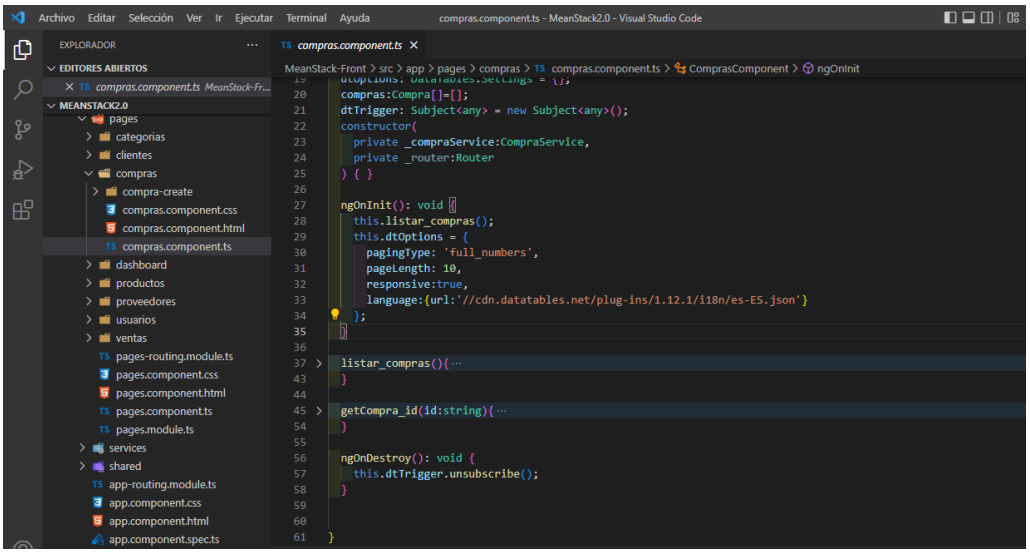

Figura 51: *Lógica de programación del componente Compras*

Elaborado por: *El autor, 2022*

Finalmente, se visualiza en la Figura 52, la vista principal del componente de compras.

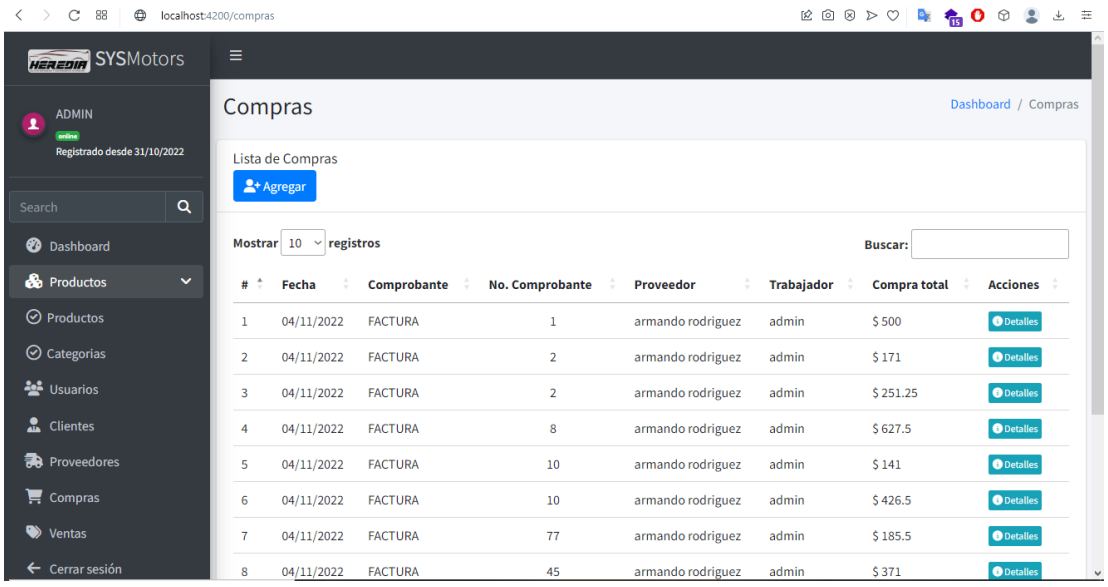

Figura 52: *Vista principal del componente Compras.*

Elaborado por: *El autor, 2022*

Desarrollo del componente Ventas

El componente de ventas al igual que el de compras y el de productos utilizó el patrón Maestro-Detalle, su estructura es similar a la del componente Compras con la diferencia del registro de clientes, en reemplazo del registro de proveedores. El maquetado se realizó tomando en cuenta este aspecto, se puede observar en la Figura 53.

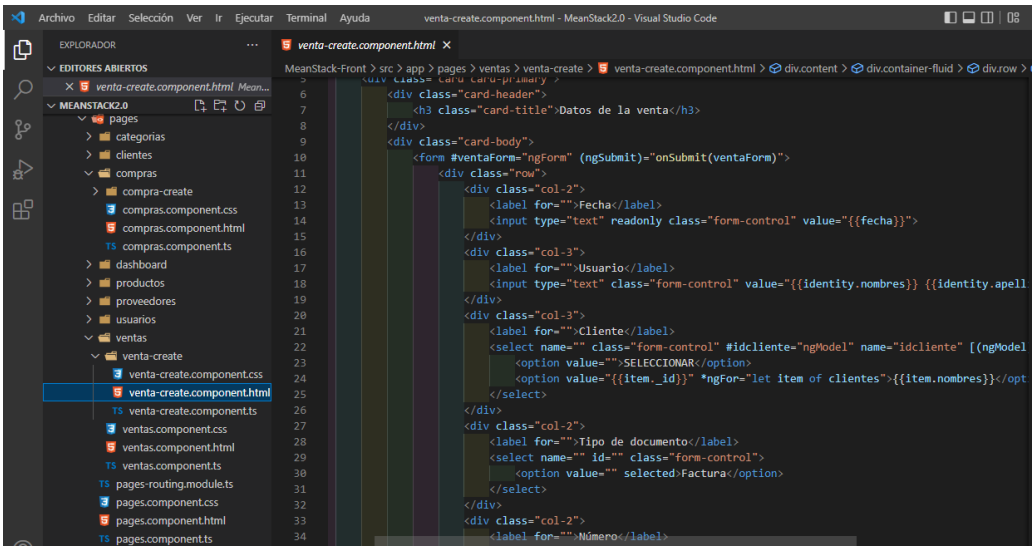

Figura 53: *Maquetación del componente de Ventas.*

Elaborado por: *El autor, 2022*

En la lógica de programación, al igual que en el componente de compras, no se puede eliminar los registros de ventas, esto con el fin de generar reportes que sirvan de evidencia para el pago de impuestos. En la Figura 54, se observa la implementación de la lógica para el componente de Ventas.

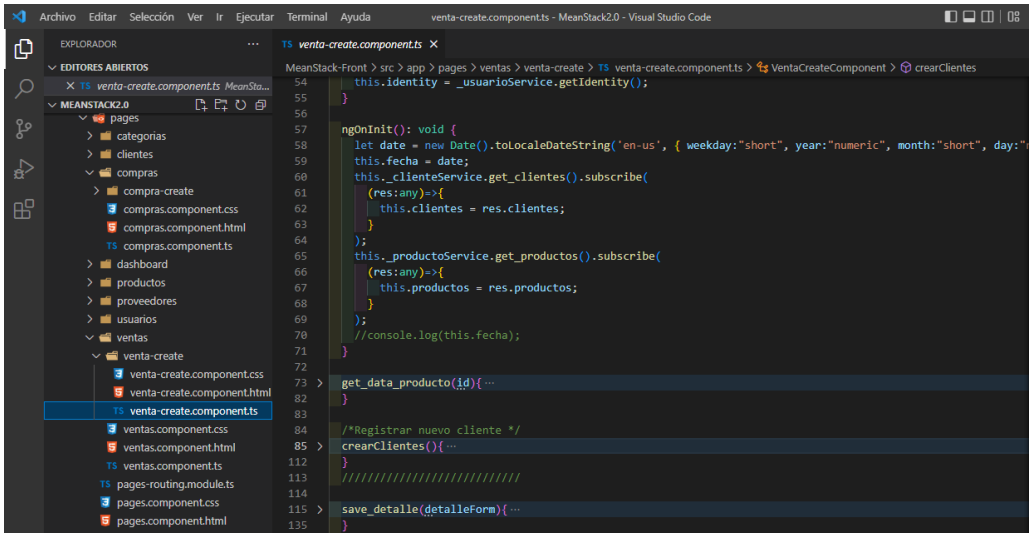

Figura 54: *Lógica de programación del componente Ventas*

Elaborado por: *El autor, 2022*

Finalmente, en la Figura 55, se muestra la vista principal del componente de Ventas.

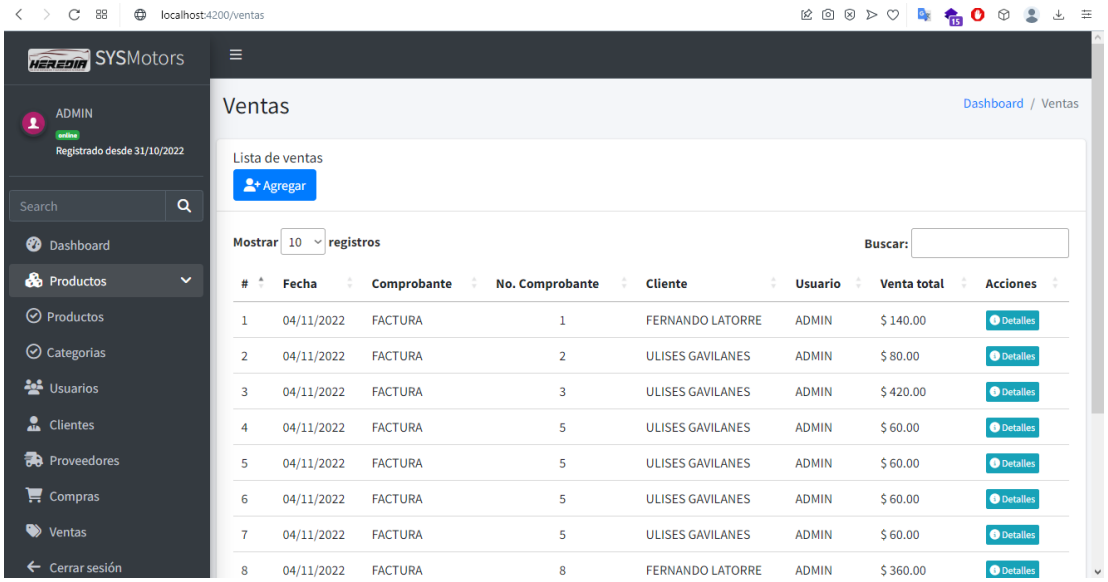

Figura 55: *Vista principal del componente de Ventas*

Elaborado por: *El autor, 2022*

Desarrollo del componente Reporte de compras

Este componente muestra el registro diario de compras realizadas, toma los datos de los componentes: Productos, Compras, Proveedores y Usuarios, para mostrar una tabla con la información necesaria para la gestión de compras. En la Figura 56, se aprecia el código implementado para la visualización de esta consulta.

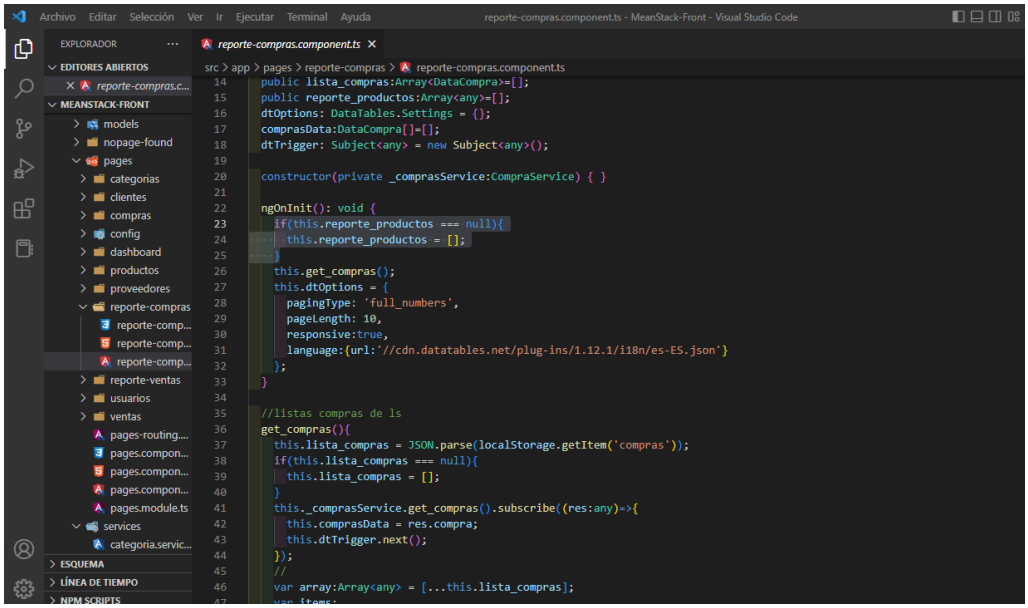

Figura 56: *Lógica de programación del componente Reporte de compras*

Elaborado por: *El autor, 2022*

En la Figura 57, se presenta el módulo desarrollado en la vista del sistema web, como se mencionó, este módulo ocupa consultas de los diferentes módulos del sistema, al final de la tabla aparece una fila con la sumatoria de cada columna.

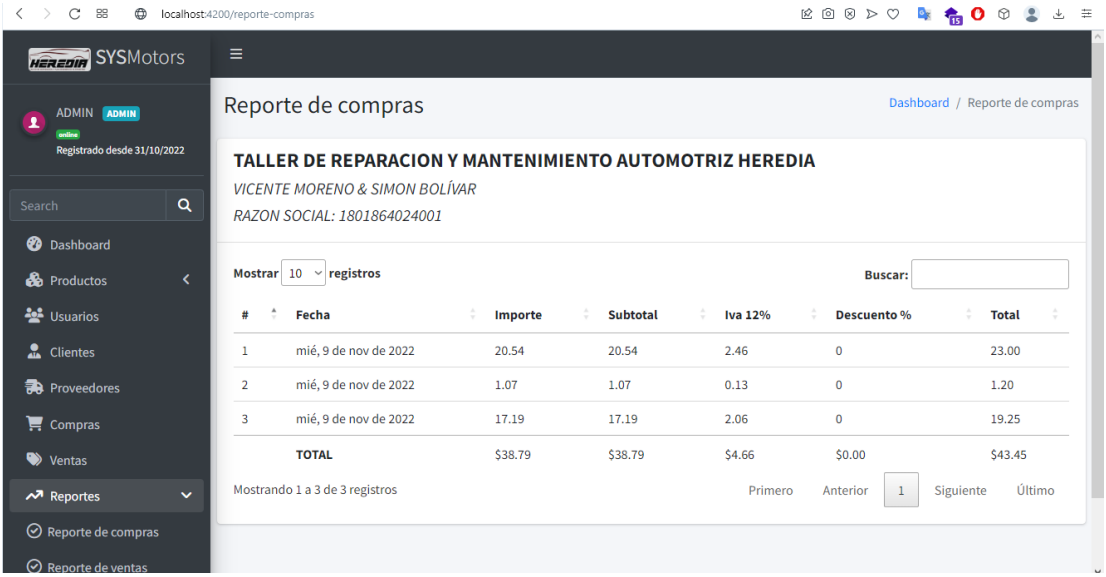

Figura 57: *Vista del reporte de compras*

Elaborado por: *El autor, 2022*

Desarrollo del componente Reporte de Ventas

Este componente muestra una consulta de los diferentes módulos del sistema web en una tabla, este módulo tiene la finalidad de gestionar las ventas que se realizan diariamente, al final de la tabla se realiza un cálculo automático de los valores de cada columna, se muestra en la Figura 58 el código para implementar esta consulta.

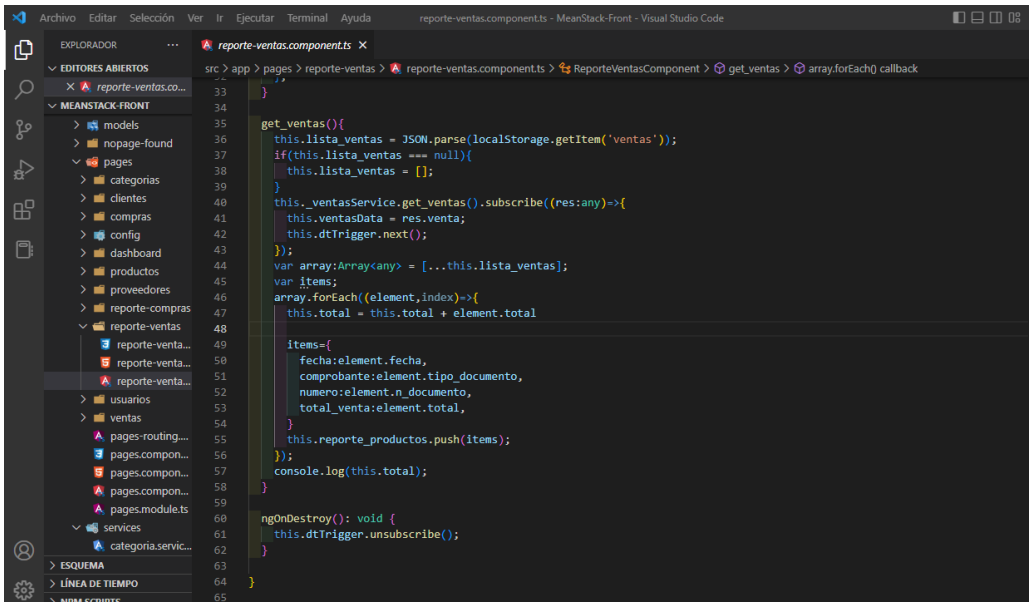

Figura 58: *Lógica de programación del componente Reporte de ventas*

Elaborado por: *El autor, 2022*

Finalmente, en la Figura 59 se presenta el módulo desarrollado en la vista principal del sistema web.

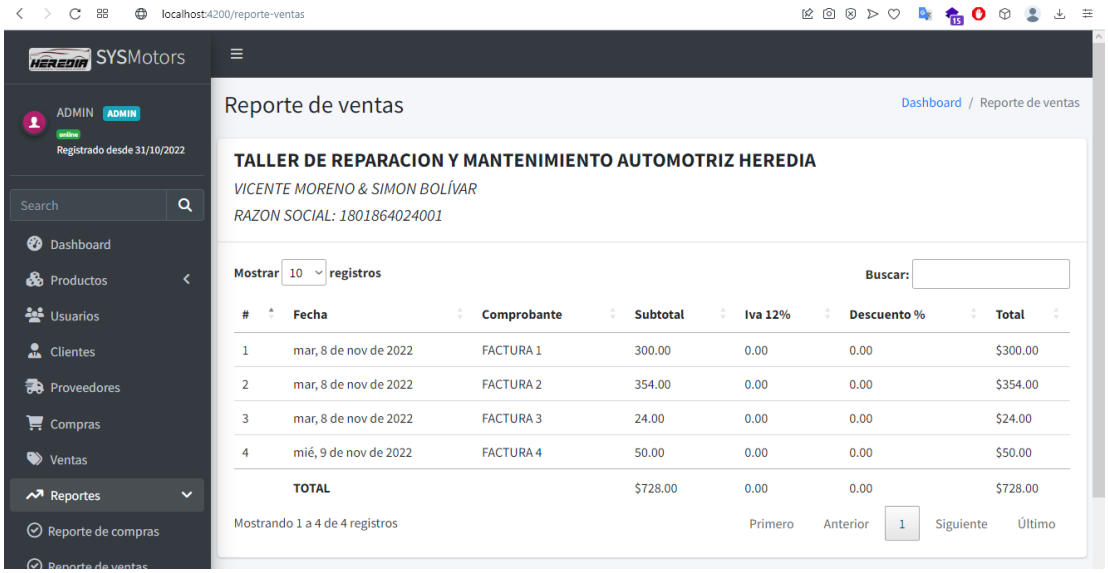

Figura 59: *Vista del reporte de ventas*

Elaborado por: *El autor, 2022*

3.5.4 Tablero Kanban Cuarta Etapa

En esta etapa se realizaron las pruebas de rendimiento del sistema web en ejecución, la herramienta utilizada es JMeter, con esta evaluación se procedió a listar los errores registrados para posterior mejoramiento.

Tabla 10: *Actividades planificadas en la cuarta etapa*

PENDIENTE	EN PROCESO	FINALIZADA
Pruebas de rendimiento del sistema web con la herramienta JMeter	Pruebas de funcionamiento del Frontend con la herramienta de desarrolladores del navegador Chrome.	Realizar la toma de requerimientos funcionales y no funcionales.
Corrección de errores en el sistema		Realizar el diseño del logo empresarial Realizar el diagrama de casos de uso del sistema web Revisar la documentación de TypeScript Revisar la documentación de Node y Express Revisar la documentación de MongoDB Revisar la documentación de Angular Revisar la documentación de Node y Express Realizar el diseño de la interfaz gráfica de usuario del sistema web con el software Pencil

	Realizar el modelado de datos con el software Erwin Data Modeler Realizar la instalación de NodeJS, ExpressJS, AngularJS. Instalar Postman y Robo 3T Desarrollo del Backend: creación del servidor, base de datos, modelos, controladores, servicios y rutas. Pruebas de funcionamiento del Backend con Postman. Desarrollo del Frontend: componentes, modelos, servicios y rutas.

Elaborado por: *El autor, 2022*

3.5.4.1 Pruebas de rendimiento del sistema web

Durante la evaluación del rendimiento del sistema web se utilizaron dos herramientas para medida de carga y estrés, y para medir el rendimiento: BlazeMeter y JMeter, con BlazeMeter se captura en un archivo de tipo. jmx un recording a través de la interfaz de usuario, después se exporta hacia JMeter para realizar las pruebas de rendimiento y carga al servidor. En la Figura 60, se observa como BlazeMeter captura por medio de una grabación los eventos que suceden al ingresar al sistema.

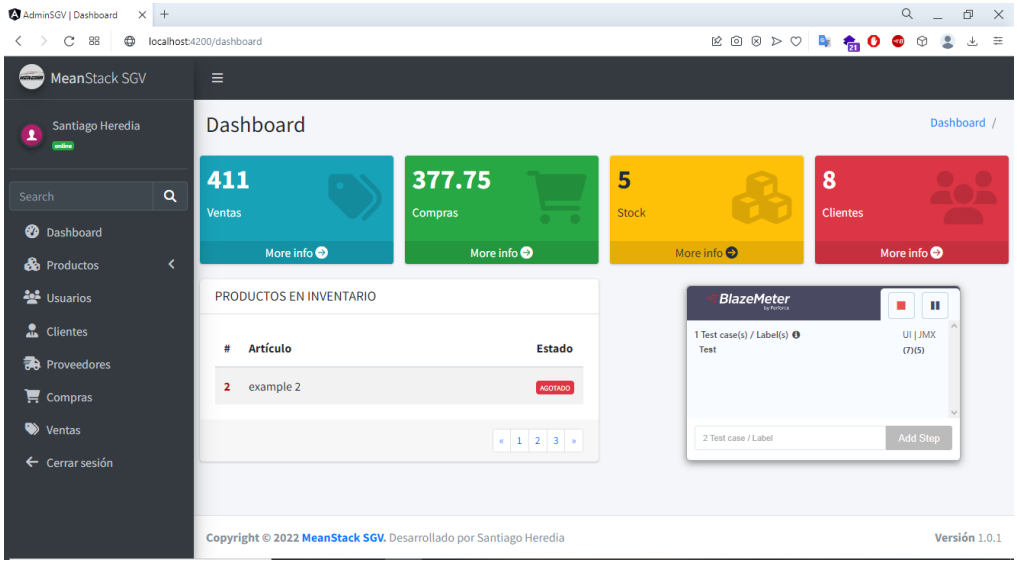

Figura 60: *Captura de eventos con BlazeMeter*

Elaborado por: *El autor, 2022*

Se procede abrir la descarga del archivo capturado en JMeter, en la Figura 61, se observa las peticiones que se realizan al servidor después de pasar por el filtro del login, también se observa la implementación de 3 escuchas de JMeter: árbol de resultados, reporte final resumido y gráfico de resultados, con estas herramientas se conoce el rendimiento del sistema en ejecución.

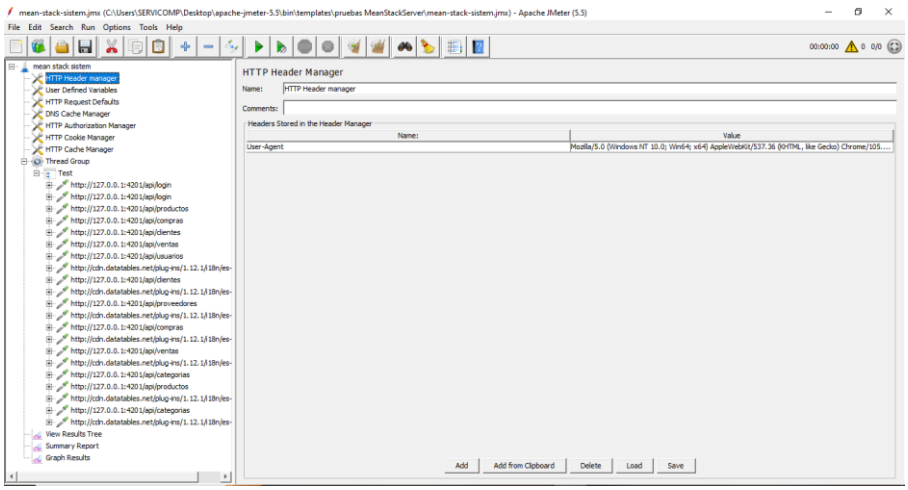

Figura 61: *Eventos capturados por BlazeMeter*

Elaborado por: *El autor, 2022*

Tiempo de respuesta

Este oyente de JMeter se caracteriza por mostrar los resultados de solicitud al servidor, procesamiento de datos y devolución de respuesta al cliente.

Pruebas de carga y estrés

Estas pruebas consistieron en evaluar de manera individual la interacción de cada módulo con el servidor, de esta manera se detallaron en un informe resumido que proporciona el oyente de JMeter métricas como: cantidad de threads (hilos o peticiones), promedio de retorno, tiempo mínimo y máximo en el envío de datos, promedio de desviación, rendimiento, cantidad de kilobytes enviados y recibos y, promedio de error en cada petición.

3.5.5 Tablero Kanban Quinta Etapa

En la Tabla 11, se muestra la actualización de actividades planificadas en el desarrollo del sistema web, se procede a ubicar en la columna finalizada las actividades completadas y se añaden las últimas actividades a realizar en el desarrollo de la investigación.

Tabla 11: *Actividades planificadas en la quinta etapa*

PENDIENTE	EN PROCESO	FINALIZADA
	Pruebas de rendimiento del sistema web con la herramienta JMeter	Realizar la toma de requerimientos funcionales y no funcionales.
	Corrección de errores en el sistema	Realizar el diseño del logo empresarial
		Realizar el diagrama de casos de uso del sistema web
		Revisar la documentación de

		TypeScript Revisar la documentación de Node y Express Revisar la documentación de MongoDB Revisar la documentación de Angular Revisar la documentación de Node y Express Realizar el diseño de la interfaz gráfica de usuario del sistema web con el software Pencil Realizar el modelado de datos con el software Erwin Data Modeler Realizar la instalación de NodeJS, ExpressJS, AngularJS. Instalar Postman y Robo 3T Desarrollo del Backend: creación del servidor, base de datos, modelos, controladores, servicios y rutas. Pruebas de funcionamiento del Backend con Postman. Desarrollo del Frontend: componentes, modelos, servicios y rutas. Pruebas de funcionamiento del Frontend con la herramienta de desarrolladores del navegador Chrome. Pruebas de rendimiento del sistema web con la herramienta JMeter

Elaborado por: *El autor, 2022*

3.5.6 Tablero Kanban Sexta Etapa (Final)

Finalmente, con la evaluación final del rendimiento del sistema se ubican las actividades que posteriormente se encontraban en proceso hacia la columna de finalizada, dando por hecho la finalización y pruebas de rendimiento del sistema web con el framework Mean Stack para la gestión de ventas.

Tabla 12: *Actividades planificadas en la sexta etapa*

PENDIENTE	EN PROCESO	FINALIZADA
		Realizar la toma de requerimientos funcionales y no funcionales. Realizar el diseño del logo empresarial Realizar el diagrama de casos de uso del sistema web Revisar la documentación de TypeScript Revisar la documentación de Node y Express Revisar la documentación de MongoDB Revisar la documentación de Angular

	Revisar la documentación de Node y Express
	Realizar el diseño de la interfaz gráfica de usuario del sistema web con el software Pencil
	Realizar el modelado de datos con el software Erwin Data Modeler
	Realizar la instalación de NodeJS, ExpressJS, AngularJS.
	Instalar Postman y Robo 3T
	Desarrollo del Backend: creación del servidor, base de datos, modelos, controladores, servicios y rutas.
	Pruebas de funcionamiento del Backend con Postman.
	Desarrollo del Frontend: componentes, modelos, servicios y rutas.
	Pruebas de funcionamiento del Frontend con la herramienta de desarrolladores del navegador Chrome.
	Pruebas de rendimiento del sistema web con la herramienta JMeter
	Pruebas de rendimiento del sistema web con la herramienta JMeter
	Corrección de errores en el sistema

Elaborado por: *El autor, 2022*

CAPÍTULO IV

4 RESULTADOS Y DISCUSIÓN

4.1 Resultados

4.1.1 Tiempo de respuesta

Se realizó la prueba de tiempo de respuesta del servidor, con una carga de 100 usuarios en un periodo de 60 segundos por petición, se observó que los servicios de login, compras y productos llegan a alcanzar un pico máximo de 60 segundos, esto debido a que el sistema debe cargar librerías y enlaces externos para la representación de resultados en tablas, en este caso se usó la librería externa de Angular Data Tables. Sin embargo, esto no provoca que el servidor deje de enviar respuestas correctas ya que incluso con una carga hostil de 100 usuarios, es capaz de resolver todas las peticiones y mantenerlas dentro de un margen estable de 30 segundos.

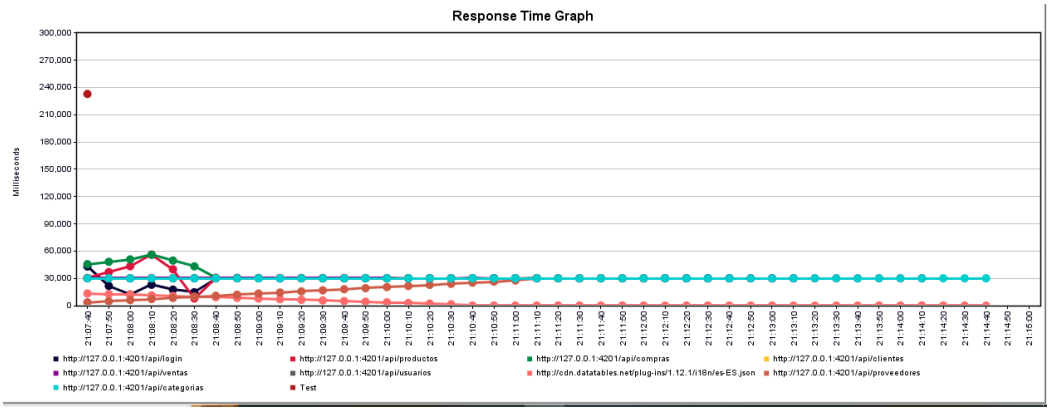

Figura 62: *Prueba de tiempo de respuesta con 100 usuarios*

Elaborado por: *El autor, 2022*

En la Figura 62, se muestra la gráfica del tiempo de respuesta del sistema con una sobrecarga de 100 usuarios, en el eje X se muestra el tiempo transcurrido en el que se resolvieron las 100 peticiones, cada petición tiene un lapso de 10 segundos; y en el eje Y el tiempo de respuesta por cada petición representado en milisegundos.

4.1.2 Carga y estrés

Finalmente, para la prueba de carga y estrés se evaluó independientemente cada servicio con una carga de 1, 10, 20, 30 threads, el objetivo de esta prueba fue mostrar los márgenes de error de cada petición al servidor, así como el rendimiento, kilobytes enviados y recibidos, desviación y promedio de respuesta. A continuación, se muestran los resultados de las pruebas de carga y estrés.

Tabla 13: *Prueba de rendimiento con 1 thread.*

Etiqueta	Muestra	Prom. (ms)	Tiempo (Min.)	Tiempo (Max.)	Desviación	Rendimiento (petición/seg)	Sent Kb/s	Rec. Kb/s	Error (%)
Login	2	600	496	705	104.50	1.7	0.81	1.96	0
Productos	2	57	32	83	25.50	3.7	1.61	5.34	0
Categorías	2	21	17	25	4.00	13.5	5.42	11.09	0
Ventas	2	23	22	25	1.50	6.3	2.71	11.11	0
Clientes	2	7	4	10	3.00	10.4	4.46	12.14	0
Compras	2	32	15	49	17.00	6.1	2.61	14.99	0
Usuarios	1	13	13	13	0.00	76.9	28.77	126.43	0
Proveedor	1	4	4	4	0.00	250.0	94.24	286.38	0
TOTAL	22	163	4	1800	394.93	12.1	9.49	35.25	0

Elaborado por: *El autor, 2022*

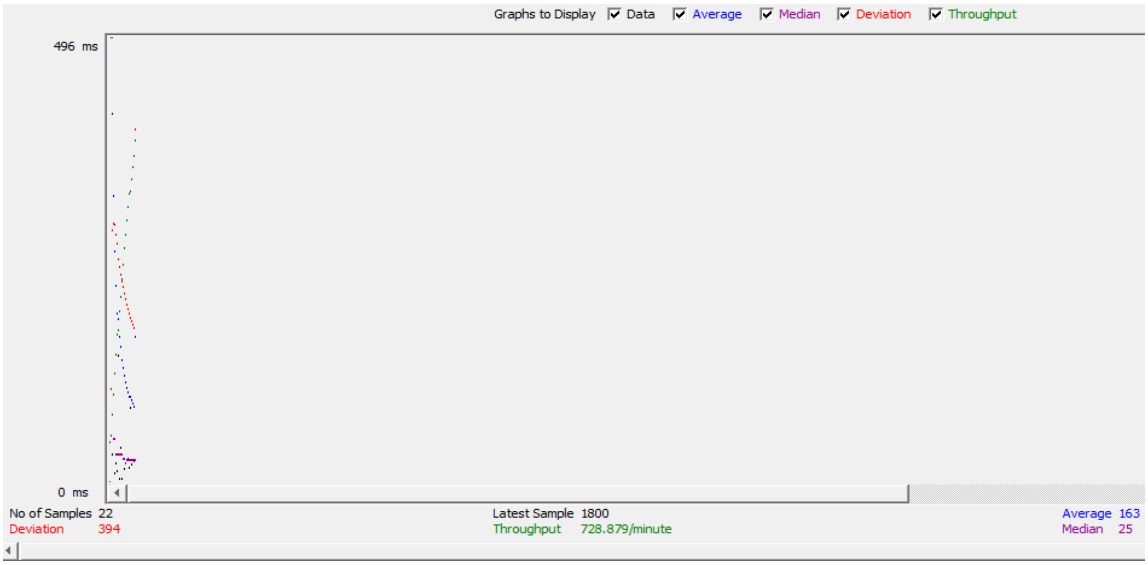

Figura 63: *Gráfica comparativa entre rendimiento y desviación con 1 usuario*

Elaborado por: *El autor, 2022*

Como se puede apreciar en la Tabla 13, el rendimiento del sistema web con 1 carga es de 12.1 peticiones por segundo con un margen de error del 0%, de igual manera los resultados obtenidos en la gráfica mostrados en la Figura 63, indican que el rendimiento, en color verde, y la desviación en color rojo se aproximan ligeramente.

Tabla 14: *Prueba de rendimiento con 10 threads.*

Etiqueta	Muestra	Prom. (ms)	Tiempo (Min.)	Tiempo (Max.)	Desviación	Rendimiento (petición/seg)	Sent Kb/s	Rec. Kb/s	Error (%)
Login	20	5050	942	13539	3347.01	1.2	0.60	1.46	0
Productos	19	2161	119	11932	3178.37	1.1	0.47	1.60	0
Categorías	20	179	51	421	110.98	17.3	6.94	14.20	0
Ventas	20	506	121	775	205.77	6.6	2.83	11.60	0
Clientes	20	319	132	553	131.65	8.9	3.82	10.39	0
Compras	20	974	211	2598	844.07	4.0	1.73	9.94	0
Usuarios	10	308	121	550	123.75	7.6	2.83	12.43	0
Proveedor	10	150	108	256	40.77	14.6	5.49	16.70	0
TOTAL	218	1663	18	19754	4197.30	10.9	8.12	30.23	0

Elaborado por: *El autor, 2022*

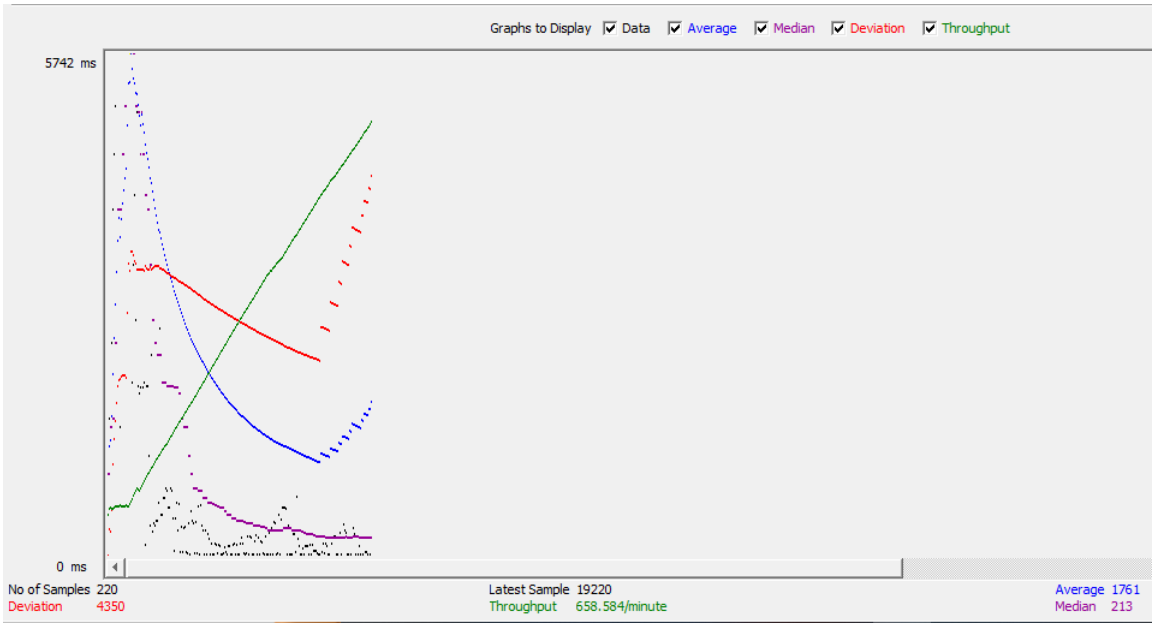

Figura 64: *Gráfica comparativa entre rendimiento y desviación con 10 usuarios*

Elaborado por: *El autor, 2022*

En la Tabla 14, se muestran los resultados obtenidos al poner a prueba el sistema web con una carga de 10 usuarios, el rendimiento obtenido es de 10.9 peticiones por segundo con un margen de error del 0%. En la Figura 64 se muestra de mejor manera como el rendimiento sobrepasa a la desviación.

Tabla 15: *Prueba de rendimiento con 20 threads*

Etiqueta	Muestra	Prom. (ms)	Tiempo (Min.)	Tiempo (Max.)	Desviación	Rendimiento (petición/seg)	Sent Kb/s	Rec. Kb/s	Error (%)
Login	28	7568	893	14937	3707.45	1.3	0.65	1.57	0
Productos	39	2658	84	7288	2863.17	3.0	1.27	4.30	0
Categorías	40	338	4	570	182.50	18.3	7.33	14.99	0
Ventas	40	600	488	12.19	126.98	7.6	3.24	13.29	0
Clientes	40	387	5	1029	225.64	11.0	4.73	12.88	0
Compras	40	737	201	7147	1038.51	3.2	1.38	7.93	0
Usuarios	20	452	270	553	89.16	12.02	4.56	20.06	0
Proveedor	20	250	6	1006	243.48	11.0	4.15	12.60	0
TOTAL	414	1471	4	26709	4027.16	15.1	8.24	30.92	0

Elaborado por: *El autor, 2022*

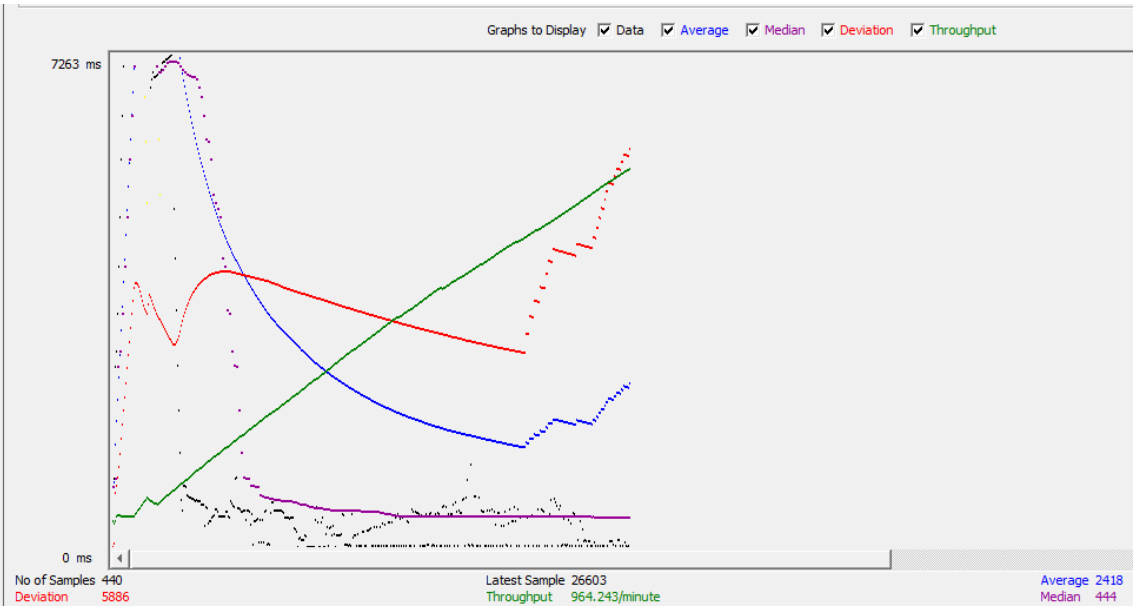

Figura 65: *Gráfica comparativa entre rendimiento y desviación con 20 usuarios*

Elaborado por: *El autor, 2022*

La tabla 15, muestra los resultados con una sobrecarga de 20 usuarios, el rendimiento total es de 15.1 peticiones por segundo y un margen de error del 0%, en la Figura 65 se puede apreciar como la desviación empieza a sobrepasar ligeramente al rendimiento.

Tabla 16: *Prueba de rendimiento con 30 threads.*

Etiqueta	Muestra	Prom. (ms)	Tiempo (Min.)	Tiempo (Max.)	Desviación	Rendimiento (petición/seg)	Sent Kb/s	Rec. Kb/s	Error (%)
Login	42	10171	522	20481	5202.42	1.6	0.76	1.83	0
Productos	57	3809	138	15629	5168.67	2.3	1.00	3.44	0
Categorías	60	422	3	943	233.15	22.6	9.06	18.52	0
Ventas	60	973	672	1710	171.57	8.9	3.81	15.60	0
Clientes	60	590	353	832	147.21	13.1	5.62	15.30	0
Compras	60	1230	574	3588	746.13	6.6	2.83	16.22	0
Usuarios	30	639	366	829	136.46	11.1	4.15	18.22	0
Proveedor	30	506	355	756	100.67	12.0	4.53	13.75	0
TOTAL	618	1920	3	34326	5128.83	17.7	9.32	35.03	0

Elaborado por: *El autor, 2022*

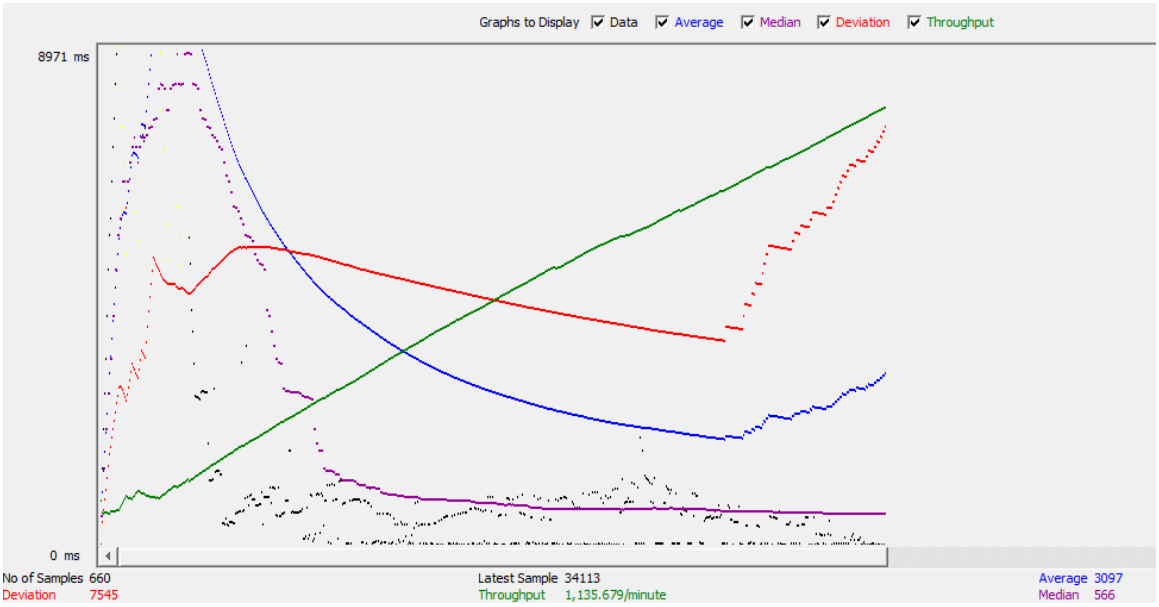

Figura 66: *Gráfica comparativa entre rendimiento y desviación con 30 usuarios*

Elaborado por: *El autor, 2022*

En la Tabla 16, se presenta los resultados del sistema con una sobrecarga de 30 usuarios, con un rendimiento total de 17.7 peticiones por segundo y un margen de error del 0%. En la Figura 66 se comprueba que el rendimiento ya sobrepasa a la desviación y se mantienen de manera creciente.

4.2 Discusión

Como ya se mencionó anteriormente, con Mean Stack se puede desarrollar aplicaciones o sistemas web completamente escalables, modernas y eficientes, además, el solo hecho de utilizar un solo lenguaje de programación para el lado del cliente y servidor ya se vuelve una idea tentativa para los desarrolladores, en especial quienes se dedican a tiempo completo al desarrollo Full Stack. En este proyecto se demostró que con Mean Stack se puede crear un sistema web para la gestión administrativa de un negocio en crecimiento, con la capacidad de almacenar y tratar gran cantidad de información y mantenerla disponible a todo momento, aparte de tener un rendimiento óptimo para la sobrecarga masiva de información.

Con precedentes de los autores (Sayago & Revelo, 2022) Mean Stack es el mejor marco de desarrollo para aplicaciones web SPA, tanta es la popularidad que incluso ya se imparte dentro de la carrera de Tecnologías de la Información en la Pontificia Universidad Católica del Ecuador. Según los investigadores en su caso de estudio, aplicación web para la gestión docente, destaca que el uso de una base de datos no relacional como MongoDB facilita la comprensión de los datos que se almacenan en formato BSON y JSON, la información resultante es legible y facilita la interpretación por parte del usuario, sin embargo, también resalta que en su caso de estudio sería interesante que se evaluara cómo reacciona dicha

aplicación web a un mayor procesamiento de datos que lleguen incluso a exigir medición de recursos del CPU y RAM de un ordenador.

De igual manera, según los autores (Dunka, Emmanuel, & Oyerinde, 2018) la integración de cuatro tecnologías (Mongo, Express, Angular y Node) dentro de un mismo Stack ya supone un mayor beneficio en términos rentabilidad y experiencia en desarrolladores, los beneficios son más altos comparados a otras tecnologías de desarrollo. En el caso de LAMP (Linux, Apache, MySql y PhP) no se puede negar que han sido pieza clave fundamental en la construcción de aplicaciones web increíbles, sin embargo, al tratarse de desarrollo web no cabe duda de que JavaScript es el lenguaje más apropiado para el trabajo. Mean Stack se ha convertido en una opción confiable por su escalabilidad en aplicaciones web en tiempo real, las tareas asíncronas que NodeJS y Express resuelven concurrentemente benefician al consumo intensivo de datos, por otro lado, MongoDB demuestra tener una gran capacidad de flexibilidad y rendimiento al proporcionar información en un formato único BSON-JSON con una fácil interpretación de la información, y por último, pero no menos importante, está Angular con un concepto arquitectónico único MVVM para la construcción de aplicaciones web SPA.

Con estos antecedentes se muestran las siguientes ventajas en el desarrollo de un sistema web de gestión de ventas con el framework Mean Stack:

- El desarrollo se simplifica al utilizar un solo lenguaje de programación para el cliente y el servidor, por lo tanto, facilita el desarrollo de proyecto administrativos para PyMES.
- La base de datos MongoDB es útil para el almacenamiento de gran cantidad de información y con la capacidad de presentarla en formato legible para el usuario, además, al no ser datos estructurados son más rápidos que una base de datos como MySql. De esta manera, el usuario tiene acceso seguro e inmediato a los recursos disponibles en el inventario del almacén, lo que lleva a solucionar uno de sus inconvenientes, la falta de control de mercadería.
- Angular es un framework que facilita el proceso de desarrollo de una interfaz interactiva, responsiva y de fácil mantenimiento, lo que beneficia a la creación de nuevos módulos para crecimiento del sistema web, en futuras versiones se podría implementar módulos de facturación electrónica, métodos de pago online, chat en tiempo real, entre otros que propicien un buen crecimiento económico y logístico en el almacén.

CAPITULO V

5 CONCLUSIONES y RECOMENDACIONES

CONCLUSIONES

- En base a la investigación realizada se concluye que el framework Mean Stack es una herramienta poderosa para el desarrollo de aplicaciones y sistemas web sostenibles y escalables, con gran capacidad a la alta concurrencia de datos gracias a una base de datos no relacional, con una comunicación asíncrona a través de peticiones HTTP lo que posibilita trabajar por separado la interfaz de usuario y el servidor, además, la ventaja de utilizar únicamente JavaScript como lenguaje de programación, convierten a Mean Stack una vía de trabajo factible para desarrolladores Full Stack.
- Se logró desarrollar un API Rest con la ayuda de Express y NodeJS, con la capacidad de resolver correctamente las peticiones HTTP hacia la base de datos en MongoDB, esto para poder resolver las actividades comerciales del almacén, en especial mantener un control sobre las existencias, gestión de entradas y salidas de mercadería, información mensual de ingresos y egresos, además de una correcta comunicación asíncrona con la interfaz gráfica desarrollada con Angular, de este modo se logra tener un control más interactivo del sistema web y, por lo tanto, ofrecer una mejor atención al cliente.
- La evaluación del rendimiento del sistema web se logró satisfactoriamente con las herramientas BlazeMeter y JMeter, a pesar de que el sistema es de uso particular del almacén, se evaluó el rendimiento basado en el tiempo de respuesta, índices de error, promedio y desviación cuando este se ve expuesto a una carga máxima de estrés.

RECOMENDACIONES

- Se recomienda para el desarrollo de aplicaciones o sistemas web con el framework Mean Stack, utilizar los parámetros del ECMASrcipt en su versión actual, así como el uso de TypeScript tanto en el servidor como en el cliente, esto servirá de ayuda al momento de codificar ya que TypeScript al ser un lenguaje de tipado estricto no permite compilar el código en el navegador si este presenta errores o declaración de variables inutilizables, esto minimiza el tiempo de desarrollo y mantiene un código limpio.
- Mantener una correcta actualización del Api Rest es fundamental ya que un sistema web de gestión empresarial o cualquiera que se desarrollase con Mean Stack siempre estará en constante evolución y crecimiento, por lo que, se recomienda antes de actualizar los métodos, funciones o parámetros del Api guardar una copia

de seguridad de la versión anterior, de esta forma no se detendrá el flujo de actividades del sistema web.

- La herramienta de evaluación JMeter es indispensable para conocer el flujo de evolución de un sistema web, ofrece una gran cantidad de recursos para evaluar cada uno de los componentes de un servidor, para la evaluación del sistema web desarrollado solo se utilizaron 3 oyentes indispensables para el análisis de resultados, sin embargo, para futuras evaluaciones del rendimiento se recomienda evitar el uso de la interfaz de JMeter y utilizar la línea de comandos, con el fin minimizar el uso de recursos físicos que ralenticen el ordenador y provoquen algún tipo de resultado inesperado.

BIBLIOGRAFÍA

Bravo, L. (2018). *Framework o librerías: ventajas y desventajas*. Obtenido de tiThink Technology: https://www.tithink.com/es/2018/08/29/framework-o-librerias-ventajas-y-desventajas/

Canós , J., Letelier, P., & Penadés, M. (2003). Métodologías Ágiles en el Desarrollo de Software. *Universidad Politecnica de Valencia*, 1-8.

Chan, J., Chung , R., & Huang, J. (2019). *Python API Development Fundamentals: Develop a full-stack web application with Python and Flask.* Birmingham: Packt Publishing.

Ciceri, M. (2019). *Introducción a Laravel: Aplicaciones robustas y a gran escala.* Bueno Aires: Creative Andina Corp.

Dunka, B., Emmanuel, E., & Oyerinde, D. (2018). Simplifying Web Application Development Using - Mean Stack. *International Journal of Latest Research in Engineering and Technology, Vol. 4*, pp. 62-76. Obtenido de https://www.researchgate.net/profile/Yinka-Oyerinde/publication/322821888_Simplifying_Web_Application_Development_Using-Mean_Stack_Technologies/links/5a71a9fba6fdcc33daaa9465/Simplifying-Web-Application-Development-Using-Mean-Stack-Technologies.pdf

Gaete et al. (2021). Enfoque de aplicación ágil con Serum, Lean y Kanban. *Ingeniare. Revista chilena de ingeniería*, 141-157. doi:10.4067/s0718-33052021000100141

Gaete, J., Villaroel, R., Figueroa, I., Cornide-Reyes, H., & Muñoz, R. (2021). Enfoque de aplicación ágil con Serum, Lean y Kanban. *Ingeniare. Revista chilena de ingeniería*, 141-157. doi:10.4067/s0718-33052021000100141

Gallegos, C. (2016). *Análisis del aporte de la implementación de un sistema integrado de información "ERP", en el mejoramiento de la gestión administrativa financiera de las empresas (tesis de maestría)*. Obtenido de Universidad Andina Simón Bolívar: https://repositorio.uasb.edu.ec/bitstream/10644/4953/1/T1934-MBA-Gallegos-Analisis.pdf

Holmes, S. (2019). *Getting MEAN with Mongo, Express, Angular, and Node 1st Edición.* Manning Publications.

Medrano, S. (2020). *Las Pymes se reactivan con negocios digitales*. Retrieved from Vistazo: https://www.vistazo.com/enfoque/las-pymes-se-reactivan-con-negocios-digitales-MEVI205966

Mejía, M., Haviv, A., & Onodi, R. (2017). *Web Application Development with MEAN*. Birmingham: Packt Publishing.

Muente, G. (2020). *Guía completa del Framework: qué es, cuáles tipos existen y por qué es importante en Internet*. Obtenido de Rock Content - ES: https://rockcontent.com/es/blog/framework/

Muradas, Y. (2022). *Qué es Postman y primeros pasos*. Obtenido de Open Webinars: https://openwebinars.net/blog/que-es-postman/

Padilla et al. (2020). Análisis a la políica tributaria del sector automotriz de la provincia de Chimborazo. *Dominio de las Ciencias*, 464-483.

Puciarelli, L. (2020). *Node JS - Vol.1: Instalación - Arquitectura - node y npm.* Bueno Aires: Creative Andina Corp.

Robledano, A. (2021). *Qué es Javascript*. Obtenido de OpenWebinars: https://openwebinars.net/blog/que-es-javascript/

Rosoff, J. (2018). *Master Detail Transactions in MongoDB*. Retrieved from MongoDB: https://www.scribbr.es/detector-de-plagio/generador-apa/new/webpage/

Salvaggio, A., & Testa, G. (2019). *JavaScript: Guía completa.* Marcombo.

Sayago, J., & Revelo, F. (2022). Aplicaciones webmodernas con stack MEAN: Un caso de estudio. *Revista Killkana Técnica. Vol. 6, No. 1,, Vol.6*, pp. 31-42. Obtenido de https://doi.org/10.26871/killkanatecnica.v6i1.989

Singh, A., & Ahmad, S. (2019). *MongoDB Simply In Depth.* Independently Published.

Soni, R. (2018). *Full Stack Angularjs for Java Developers: Build a Full-Featured Web Application from Scratch Using Angularjs with Spring Restful.* Bangalore: Apress.

UNIR. (2022). *MERN y MEAN Stack, las agrupaciones de herramientas de software más populares*. Obtenido de UNIR: https://www.unir.net/ingenieria/revista/mern-vs-mean-stack/

Urtiaga, G. (2020). *Administrar MySQL y MariaDB: Aprende a administrar MySQL y MariaDB fácilmente.*

Uzayr, S. (2022). *Getting the Most Out of Node.Js Frameworks: The Essential Tools and Libraries.* CRC Press.