



**UNIVERSIDAD POLITÉCNICA SALESIANA**

**SEDE QUITO**

**CARRERA DE COMPUTACIÓN**

**DESARROLLO DE UN SISTEMA DE ADMINISTRACIÓN DE VENTAS Y CONTROL  
DE INVENTARIOS PARA TIENDAS EN LÍNEA. CASO DE ESTUDIO: INVIRTUAL  
STORE**

Trabajo de titulación previo a la obtención del  
Título de Ingenieros en Ciencias de la Computación

AUTORES: EDWIN ALFONSO HERNÁNDEZ MOYA

JEFRY ALEXANDER NAVAS SILVA

TUTOR: RODRIGO EFRAÍN TUFÍÑO CÁRDENAS

Quito – Ecuador

2023

## **CERTIFICADO DE RESPONSABILIDAD Y AUDITORÍA DEL TRABAJO DE TITULACIÓN**

Nosotros Edwin Alfonso Hernández Moya con documento de identificación N° 1722122783 y Jefry Alexander Navas Silva con documento de identificación N° 1719591719 manifestamos que: Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 28 de febrero del año 2023

Atentamente,



---

Edwin Alfonso Hernández Moya  
1722122783



---

Jefry Alexander Navas Silva  
1719591719

## **CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA**

Nosotros Edwin Alfonso Hernández Moya con documento de identificación N° 1722122783 y Jefry Alexander Navas Silva con documento de identificación N° 1719591719, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto técnico: “Desarrollo de un sistema de administración de ventas y control de inventarios para tiendas en línea. Caso de estudio: Invirtual Store”, el cual ha sido desarrollado para optar por el título de: Ingenieros en Ciencias de la Computación en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 28 de febrero del año 2023

Atentamente,



---

Edwin Alfonso Hernández Moya  
1722122783



---

Jefry Alexander Navas Silva  
1719591719

## **CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN**

Yo, Rodrigo Efraín Tufiño Cárdenas con documento de identificación N° 1717646390, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DESARROLLO DE UN SISTEMA DE ADMINISTRACIÓN DE VENTAS Y CONTROL DE INVENTARIOS PARA TIENDAS EN LÍNEA. CASO DE ESTUDIO: INVIRTUAL STORE, realizado por: Edwin Alfonso Hernández Moya con documento de identificación N° 1722122783 y Jefry Alexander Navas Silva con documento de identificación N° 1719591719, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 28 de febrero del año 2023

Atentamente,



---

Ing. Rodrigo Efraín Tufiño Cárdenas Msc  
1717646390

## **AGRADECIMIENTO**

En primer lugar, nos gustaría expresar nuestro agradecimiento al tutor de esta tesis Rodrigo Efraín Tufiño por su apoyo, paciencia y conocimiento, por guiarnos en el camino para la realización de este trabajo de titulación.

A nuestros amigos y compañeros de la carrera de Ingeniería en Ciencias de la Computación. Hoy nos toca cerrar un capítulo maravilloso en esta historia de vida y no podemos dejar de agradecerles por su apoyo y perseverancia durante los momentos más difíciles, por las horas de aprendizaje juntos. Gracias por estar siempre ahí.

## ÍNDICE GENERAL

1. INTRODUCCIÓN .....	1
1.1 ANTECEDENTES .....	1
1.2 PROBLEMA .....	4
1.3 JUSTIFICACIÓN.....	7
1.4 OBJETIVOS.....	8
1.4.1 Objetivo General .....	8
1.4.2 Objetivos Específicos .....	8
1.5 METODOLOGÍA.....	9
2. ANÁLISIS Y DISEÑO .....	10
2.1 ANÁLISIS DEL PROBLEMA .....	10
2.1.1 InVirtual Store.....	10
2.1.2 Cadena de Suministro.....	11
2.1.3 Proceso del Negocio.....	13
2.1.4 Flujo de Datos del Proceso.....	18
2.2 ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE .....	22
2.2.1 Perspectiva del Producto .....	22
2.2.2 Funcionalidades del Producto .....	22
2.2.3 Ambiente de Operaciones .....	23
2.2.4 Características de cada perfil de usuario .....	24
2.2.5 Restricciones .....	25
2.2.6 Políticas .....	25
2.2.7 Limitaciones de Hardware .....	25
2.2.8 Requisitos futuros.....	25
2.2.9 Requerimientos funcionales específicos .....	26
2.2.10 Requerimientos no funcionales .....	29
2.3 DISEÑO DEL SISTEMA.....	30
2.3.1 Arquitectura del Sistema .....	31
2.3.2 Base de datos .....	35
2.3.3 Diagramas de Secuencia.....	41
2.3.4 Diseño de Interfaz .....	48
3. CONSTRUCCIÓN Y PRUEBAS .....	57
3.1 CONSTRUCCIÓN .....	57
3.1.1 Metodología de Desarrollo.....	57

3.1.2	Product Backlog .....	57
3.1.3	Sprints.....	59
3.2	PRUEBAS .....	95
3.2.1	Pruebas del Código.....	95
3.2.2	Pruebas de Rendimiento.....	98
4.	IMPLEMENTACIÓN.....	104
4.1	DIAGRAMA DE DESPLIEGUE.....	104
4.2	DESPLIEGUE DE LA BASE DE DATOS .....	105
4.3	DESPLIEGUE DE LA APLICACIÓN .....	107
4.4	DESPLIEGUE DE LA API DE GESTIÓN DE REPARTICIÓN.....	109
5.	CONCLUSIONES .....	112
6.	RECOMENDACIONES .....	114

## ÍNDICE TABLAS

<b>Tabla 1</b> Perfil de usuario Administrador .....	24
<b>Tabla 2</b> Perfil de usuario Empleado .....	24
<b>Tabla 3</b> Perfil de usuario Repartidor .....	24
<b>Tabla 4</b> Perfil de usuario Cliente .....	24
<b>Tabla 5</b> Requerimiento 01 Registro de Inventario .....	26
<b>Tabla 6</b> Requerimiento 02 Registro del Cliente .....	26
<b>Tabla 7</b> Requerimiento 03 Registro de Pedido.....	27
<b>Tabla 8</b> Requerimiento 04 Seguimiento del pedido para el administrador .....	27
<b>Tabla 9</b> Requerimiento 05 Seguimiento del pedido para el cliente.....	27
<b>Tabla 10</b> Requerimiento 06 Registro de pagos .....	28
<b>Tabla 11</b> Requerimiento 07 Visualización y pagos pendientes.....	28
<b>Tabla 12</b> Requerimiento 08 Notas de ventas.....	28
<b>Tabla 13</b> Requerimiento 09 Visualización de las comisiones de los repartidores .....	29
<b>Tabla 14</b> Tabla Producto .....	36
<b>Tabla 15</b> Tabla cliente .....	36
<b>Tabla 16</b> Tabla pedido.....	38
<b>Tabla 17</b> Tabla pagos .....	38
<b>Tabla 18</b> Tabla empleado .....	39
<b>Tabla 19</b> Tabla ganancias .....	39
<b>Tabla 20</b> Tabla proveedor.....	39
<b>Tabla 21</b> Tabla categoria_producto .....	39
<b>Tabla 22</b> Tabla genero .....	40
<b>Tabla 23</b> Tabla tipo_empleado .....	40
<b>Tabla 24</b> Tabla factura.....	40
<b>Tabla 25</b> Descripcion Product Backlog.....	58
<b>Tabla 26</b> Tabla de Product Backlog .....	58



## ÍNDICE FIGURAS

<b>Figura 1</b>	Diagrama de flujo del proceso de registro de inventario .....	14
<b>Figura 2</b>	Catálogo de Productos Invirtual Store .....	15
<b>Figura 3</b>	Diagrama de Flujo realización de un pedido .....	16
<b>Figura 4</b>	Diagrama de Flujo de la venta del producto .....	17
<b>Figura 5</b>	Diagrama de Flujo de la repartición del pedido.....	18
<b>Figura 6</b>	Diagrama de Contexto de Gestión de inventario y Control de ventas.....	21
<b>Figura 7</b>	Arquitectura del Sistema.....	34
<b>Figura 8</b>	Diagrama Relacional de la Base de Datos .....	35
<b>Figura 9</b>	Diagrama de secuencia Inicio de sesión .....	42
<b>Figura 10</b>	Diagrama de secuencia Registrar Cliente .....	43
<b>Figura 11</b>	Diagrama de secuencia Registrar Producto .....	44
<b>Figura 12</b>	Diagrama de secuencia Registrar Pedido .....	45
<b>Figura 13</b>	Diagrama de secuencia Registrar Pago.....	46
<b>Figura 14</b>	Diagrama de secuencia Registrar Nota de Venta .....	47
<b>Figura 15</b>	Diagrama de secuencia Visualización de pagos .....	48
<b>Figura 16</b>	Diseño de la Interfaz de Registro del Cliente .....	49
<b>Figura 17</b>	Diseño de Interfaz de Registro de un Producto .....	50
<b>Figura 18</b>	Diseño de Interfaz de Registro de un Pedido.....	51
<b>Figura 19</b>	Diseño de Interfaz de Asignación de Pedidos .....	52
<b>Figura 20</b>	Diseño de Interfaz de Registro de Pagos .....	53
<b>Figura 21</b>	Diseño de Interfaz de Visualización del Detalle del Pedido.....	54
<b>Figura 22</b>	Diseño de Interfaz de Creación de Nota de Venta.....	55
<b>Figura 23</b>	Diseño de Interfaz de Seguimiento de un Pedido .....	56
<b>Figura 24</b>	Tablero Kanban durante el Sprint 1 .....	61
<b>Figura 25</b>	Carga del Script Inicial .....	62
<b>Figura 26</b>	Carga de tablas con datos estáticos.....	62
<b>Figura 27</b>	Conexión local a la Base de Datos .....	64
<b>Figura 28</b>	Inicio de la Aplicación.....	64
<b>Figura 29</b>	Repositorio Github.....	65
<b>Figura 30</b>	Tablero Kanban durante el Sprint 2.....	66
<b>Figura 31</b>	Sesiones en la Aplicación .....	66
<b>Figura 32</b>	Obtención del usuario logeado .....	67
<b>Figura 33</b>	Asignación de sesiones por perfil de usuario .....	68
<b>Figura 34</b>	Rutas de la Aplicación .....	68
<b>Figura 35</b>	Controlador de Ruta de un perfil .....	69
<b>Figura 36</b>	Controlador de Ruta de varios perfiles .....	69
<b>Figura 37</b>	Pantalla Inicial del Administrador .....	70
<b>Figura 38</b>	Pantalla inicial de un empleado .....	71
<b>Figura 39</b>	Tablero Kanban durante el Sprint 3.....	72
<b>Figura 40</b>	Formulario de registro de empleado .....	73
<b>Figura 41</b>	Subida de archivos con Multer .....	74
<b>Figura 42</b>	Imagen del Producto .....	74
<b>Figura 43</b>	Abstracción del path del archivo .....	75
<b>Figura 44</b>	Path de la imagen del Producto .....	75
<b>Figura 45</b>	Formulario de registro del cliente.....	76

<b>Figura 46</b> Condicional HandleBars en HTML.....	77
<b>Figura 47</b> Plantilla HTML de la carga de productos.....	78
<b>Figura 48</b> Interfaz del Registro del Pedido .....	78
<b>Figura 49</b> Tablero Kanban durante el Sprint 4.....	79
<b>Figura 50</b> Formulario de Registro de pago .....	80
<b>Figura 51</b> Cálculo del saldo pendiente .....	80
<b>Figura 52</b> Verificación del saldo de un pedido .....	81
<b>Figura 53</b> Formulario Registro de la Nota de Venta .....	82
<b>Figura 54</b> Consulta actualización del pedido .....	83
<b>Figura 55</b> Función asíncrona Seguimiento del pedido.....	84
<b>Figura 56</b> Interfaz Seguimiento del Pedido.....	84
<b>Figura 57</b> Evaluaciones del Estado en la Interfaz .....	85
<b>Figura 58</b> Tablero Kanban durante el sprint 5 .....	86
<b>Figura 59</b> Tabla de Productos .....	87
<b>Figura 60</b> Tabla de Usuarios .....	87
<b>Figura 61</b> Tabla de Repartidores y sus Ganancias .....	88
<b>Figura 62</b> Tabla de Pedidos.....	89
<b>Figura 63</b> Detalle del pedido.....	89
<b>Figura 64</b> Interfaz Modificar Producto .....	90
<b>Figura 65</b> Interfaz Modificar Usuario .....	91
<b>Figura 66</b> Tablero Kanban durante el sprint 6 .....	91
<b>Figura 67</b> Interfaz Asignar Repartidores del día.....	92
<b>Figura 68</b> API Clustering .....	93
<b>Figura 69</b> Tablero Kanban durante el sprint 7 .....	94
<b>Figura 70</b> Alertas del Sistema .....	94
<b>Figura 71</b> Validaciones Formularios en el Sistema.....	94
<b>Figura 72</b> Gráfico de Resultado de Pruebas de código con SonarQube .....	96
<b>Figura 73</b> Problemas de seguridad por Categoría y Prueba Ejecutada .....	98
<b>Figura 74</b> Reporte de rendimiento de la ruta /pedidos .....	100
<b>Figura 75</b> Reporte de rendimiento de la ruta /tablaAsignados.....	102
<b>Figura 76</b> Reporte de rendimiento de la ruta /tusPedidos .....	103
<b>Figura 77</b> Diagrama de Despliegue.....	105
<b>Figura 78</b> Código de la Construcción de la aplicación en el archivo YAML.....	108
<b>Figura 79</b> Código de la Construcción de la aplicación en el archivo YAML.....	109
<b>Figura 80</b> Secuencia para la Implementación .....	111

## **RESUMEN**

El emprendimiento de importación y venta minorista InVirtual Store, presentaba varios problemas en la gestión del inventario de sus productos, y el control de los pagos que recibían. Basados en estas problemáticas, se desarrolló una aplicación web que permite el registro, seguimiento y control de los productos, pedidos, y pagos. El sistema fue desarrollado, empezando con la fase de Análisis y Diseño, continuando con la construcción de la aplicación a través de las tecnologías de Node.js y Handlebars.js. Una vez culminada la construcción, se ejecutaron pruebas de código y rendimiento, para finalmente realizar la implementación de la aplicación en la web mediante Azure App Services. Para su construcción se utilizó la metodología ágil SCRUM, contado con siete sprints definidos que permitieron una entrega iterativa de la aplicación. Como resultado, se entregó un software funcional que permite mejorar los procesos de la administración de ventas, y control de inventario en la tienda InVirtual Store.

## **ABSTRACT**

The import and retail venture InVirtual Store had several problems in managing the stock of its products and controlling the payments they received. Based on these problems, a web application was developed that allows the registration, monitoring and control of products, orders, and payments. The system was developed, starting with the Analysis and Design phase, continuing with the construction of the application through Node.js and Handlebars.js technologies. Once the construction was completed, code and performance tests were executed, to finally carry out the implementation of the application on the web through Azure App Services. For its construction, the agile methodology SCRUM was used, with seven defined sprints that allowed an iterative delivery of the application. As a result, functional software was delivered that allows improving the processes of sales administration, and stock control at InVirtual Store.

## **1. INTRODUCCIÓN**

Este capítulo proporciona información básica, una descripción del problema basada en un caso de estudio y objetivos con el fin de comprender el problema y para justificar el proyecto. Donde estas metas propuestas serán revisadas en conjunto a medida que avance el proyecto.

### **1.1 ANTECEDENTES**

(Indumathi et al., 2022), realizaron el trabajo: “Point of Sales and Inventory Management System”, ya que observaron problemas en la administración de los datos que genera la industria de ventas en línea en la ciudad de Karnataka-India. Su objetivo es facilitar mediante un software la administración del inventario mediante registro de productos y seguimiento de ventas, dando informes detallados, para que los administradores cumplan con la demanda de pedidos. El sistema se realizó bajo la metodología SCRUM, utilizando para su desarrollo la tecnología de Java, y MongoDB para su Base de Datos. En los resultados se mostró un sistema que devuelve una interfaz con los datos de los productos y ventas asociadas a cada uno. Concluyendo que el sistema permite el seguimiento de existencias en tiempo real, y la capacidad de generar informes, que facilitaron las decisiones administrativas en cuestión de stock.

(Buddika, 2017), Con su trabajo: “Purchase, Stock and Sales Management System for Gishani Distributors”, planteó como objetivo hacer que la empresa de compra y venta minorista de confitería “Gishani Distributors” de la ciudad de Horana-Sri Lanka deje los registros de papel y tenga un sistema que permita administrar artículos, gastos, detalles del cliente, órdenes de

compra y facturas de venta. Este sistema se desarrolló siguiendo la metodología Procesos Racional Unificado (RUP), utilizando para su desarrollo el lenguaje de programación Java con su tecnología JSP, y MySQL para su base de datos. Como conclusión, se mostró que los usuarios aceptaron correctamente el sistema, logrando mejorar su productividad, eficiencia y precisión en el registro de inventario de sus productos.

(Liu, 2022), En su trabajo: “Development and Application of Sales System Software Based on Computer Network” de la Shunde Polytechnic en Foshan-China, plantea como objetivo resolver el problema de la gestión de ventas tradicional de automóviles, mediante un sistema de venta y arrendamiento de automóviles, con una estructura que gestione a través de la red actividades desde pedidos, hasta servicio al cliente. El sistema se realizó siguiendo la metodología Ágil SCRUM, desarrollada mediante la arquitectura MVVM, utilizando en su back-end Java como lenguaje de programación y Spring Boot, mientras que en su front-end se utilizó Javascript con el framework de Vue, y MySQL para su base de datos. Como conclusión del proyecto, sus resultados fueron exitosos mostrando una mayor la facilidad y rapidez de gestión en el tiempo de respuesta en sus actividades, mostrando una mejora en su capacidad de gestión, dando una pauta para el desarrollo del paradigma de ventas de automóviles en Foshan.

(Martín Romero, 2019), En su trabajo: “Diseño e implementación de sistema de inventarios para el almacén de pinturas y ferretería Ferrecolor”, plantea como objetivo diseñar y desarrollar un sistema que ayude a automatizar, y gestionar de forma eficiente el inventario de la ferretería colombiana Ferrecolor. Este sistema siguió la metodología ágil SCRUM, utilizando JAVA EE,

como entorno de programación y SQL Server para su base de datos. Como conclusión del proyecto, se mostraron resultados exitosos, donde la automatización de procesos logró mejorar la gestión de stock del almacén, además de presentar un aumento considerable en la claridad del estado de su inventario, además de estar adecuado a las necesidades de los distintos clientes.

(Plúas Navarrete & Ponce Baque, 2018), En su trabajo publicado por la Universidad Politécnica Salesiana de Guayaquil-Ecuador, en su trabajo: “Desarrollo e implementación de un Sistema de control y Administración de ventas en la empresa Global Sumec”, plantearon como objetivo ayudar en la administración de la información creando un software para administrar sus ventas, y sistematizar los diversos tipos de datos comerciales del negocio. El sistema se realizó bajo la metodología SCRUM, utilizando la arquitectura MVC, programado con PHP y su base de datos en PostgreSQL. Como conclusión del proyecto, se mostraron resultados exitosos cumpliendo con todos los requerimientos necesarios, logrando mejorar el acceso a los datos comerciales del negocio, además de brindar al supervisor reportes detallados de las ventas, bodegas y actividades de sus empleados.

(Marchant Castelnuovo, 2012), de la Escuela Politécnica del Ejército, de Sangolquí-Ecuador, en su trabajo: “Diseño y desarrollo de un Sistema WMS para la empresa Logitec S.A.”, plantearon como objetivo mejorar las actividades de seguimiento de stock de productos, gestión de bodegas y la entrega a clientes finales mediante la implementación de un software de control de logística. El sistema se realizó siguiendo los parámetros de la metodología Microsoft Solution Framework (MSF), utilizando .NET como framework basado en el

lenguaje C# y SQL Server para su base de datos. Como conclusión del proyecto, se mostró un software altamente eficiente, que almacena de una manera organizada todos los ingresos, mostrando gráficos estadísticos de los diferentes indicadores de rendimiento de control.

## **1.2 PROBLEMA**

Los emprendimientos de venta a través de redes sociales se han visto modificados debido a la pandemia del COVID-19 han ocasionado un gran crecimiento en la demanda de clientes, derivando en que los emprendedores tengan que solventar una gran cantidad de pedidos (Fernando Montalvo-Coronel & Hipólito Orozco-Santos, 2020). Por tal motivo contar con un control en las operaciones de stock y ventas se ha vuelto fundamental, donde es necesario implementar acciones que permitan asegurar la satisfacción de la demanda de los clientes, ni llevar un exceso de oferta, con el fin de comprender de mejor manera los patrones de ventas (Luther, 2020). En base a esta problemática, los emprendimientos y PYMES, buscan soluciones tecnológicas para gestionar sus diversas áreas según sus requerimientos y necesidades. Existen varios tipos de soluciones tecnológicas que pueden ir de sistemas simples y enfocados en administrar el stock y ventas, o softwares más robustos que manejan cinco o más áreas de la empresa. (Dini et al., 2021)

A nivel mundial los Sistemas de Control y Gestión empresarial, se han convertido en un aspecto muy demandado y necesario. Según Statista Resource Center, (2022), se plantea que al final del 2022 exista un gasto aproximado de \$675 billones en este tipo de sistemas. Esto se debe a que las empresas de cualquier tamaño buscan impulsar su negocio y productividad a través de estas tecnologías que se adecúan a sus necesidades. Tal y como lo desarrollaron



Indumathi et al. (2022), crearon un sistema de administración de inventario y punto de venta en la ciudad de Karnataka, India, donde lograron brindar informes detallados y actualizados para los gerentes sobre su stock y ventas.

En América Latina desde hace varios años las PYMES, han ido aumentando su demanda en Sistemas de Control y Gestión empresarial, donde según Statista Research Department (2021), al 2021, las empresas han invertido \$8.6 billones en este tipo software, siendo México y Perú los países que más aportan. Esto debido principalmente al gran crecimiento de empresas de actividades comerciales a raíz de la pandemia del COVID-19. Tal y como sucedió en el 2019 en Colombia, donde Esneider Romero, implementó un sistema para conseguir administración de inventario de una ferretería. (Martín Romero, 2019)

En Ecuador, se muestra que en promedio las empresas o personas que proveen Sistemas de Control y Gestión empresarial facturan alrededor de \$1.3 millones anualmente, siendo un monto muy bajo respecto al resto de Latinoamérica (González Patricia, 2021). Esta problemática se da debido principalmente al desconocimiento y altos costos que empresas como Oddo y SAP presentan; donde la mayoría opta por opciones más pequeñas y personalizadas, como sucedió cuando estudiantes de la Universidad Politécnica Salesiana, generaron un sistema para controlar, administrar y sistematizar las ventas de la empresa de importación Guayaquileña, “Global Sumec”. (Plúas Navarrete & Ponce Baque, 2018)

El presente proyecto plantea responder a la siguiente problemática: ¿En qué medida contribuiría el desarrollo de un sistema de administración de ventas y control de inventarios de

la tienda InVirtual Store? InVirtual Store es un emprendimiento cuya actividad es la importación de diversos productos desde Estados Unidos para comercializarlos en Ecuador, utilizando como medio de venta las redes sociales, misma que ha evidenciado una deficiente administración en los apartados de stock y ventas, provocando conflictos con los cobros y ganancias de la tienda. Adicionalmente las deficiencias en su gestión de pagos han perjudicado las ventas y un registro fiable de deudas pendientes. Es evidente la necesidad de un software que gestione los problemas de stock y ventas de las tiendas, donde, a pesar de la existencia de varios softwares similares ya establecidos, representan una fuerte inversión para un emprendimiento en crecimiento, además de la falta de personalización de acuerdo con sus necesidades.

Se han presentado diversas causas que derivan en este problema, donde los más importantes son: i) Los administradores de la tienda cuentan con una mala gestión del inventario tanto de entrada como de salida de productos. ii) La tienda cuenta con un mal sistema registro y seguimiento de pagos tanto para la empresa como para sus clientes ya que todo es realizado de forma manual. iii) El reciente aumento de clientes, y pedidos, en conjunto la falta de presupuesto para costear un Sistema de Control y Gestión empresarial establecido.

Existen varios efectos que se han generado o pueden derivarse a raíz de este problema: i) Los administradores de la tienda tienen una falta de claridad en la información de sus productos disponibles, y los que se han agotado. ii) El registro de pagos y deudas pendientes tanto para los administradores como clientes se encuentra propenso a daños y pérdidas, dando la posibilidad de convertirse en información irrecuperable. iii) Existe una falta de cumplimiento

y satisfacción de la demanda, generando descontento de los clientes por falta de una gestión efectiva de la tienda.

Para solventar esta problemática, se ha propuesto el desarrollo de un Sistema de Administración y Control para la gestión de stock y ventas, con la finalidad de que los administradores de la tienda puedan gestionar y tener una visión actualizada del inventario en tiempo real para actualizar sus publicaciones de productos en redes sociales. Además, el sistema permitirá la visualización tanto para clientes como para los administradores de la información de pagos pendientes, y revisión del estado actual de los pedidos realizados.

### **1.3 JUSTIFICACIÓN**

En la actualidad, los emprendimientos de ventas a través de redes sociales han tenido un crecimiento exponencial. Presentan la gran ventaja de publicitar y llegar de forma rápida a una gran cantidad de clientes potenciales, impulsando un incremento en el número de ventas realizadas (Ochoa, 2021). Debido a esto, emprendimientos como InVirtual Store, ha presentado problemas para sus administradores en el manejo de inventario de sus productos, además de la necesidad de mejorar su gestión de pagos, ya que todo lo registran de forma manual con notas en libretas, que han generado conflictos al no tener respaldos ni rapidez de actualización de deudas pendientes tanto para la tienda, como a los clientes finales.

El presente proyecto proporcionará la creación de un sistema para administrar y controlar el stock y las ventas, de tal manera que gestione la información de los clientes y empleados a fin

de optimizar los procesos de la tienda. Para la realización de este sistema es esencial el uso de nuevas tecnologías, lo que permitirá controlar y mejorar la administración de la tienda.

Puesto que la empresa maneja información de clientes y empleados, el sistema debe contener un mejor control de ventas, y actualización del stock, de igual manera se estima una concurrencia simultánea de usuarios en la aplicación. Debido a esto es necesario el uso de un entorno de ejecución asíncrono con una tecnología emergente como lo es Node.js y Handlebars.js, los cuales optimizan los recursos de la aplicación. Logrando mejorar la administración de la información, obteniendo un crecimiento de su negocio y cartera de clientes.

## **1.4 OBJETIVOS**

### ***1.4.1 Objetivo General***

Desarrollar un sistema de administración de ventas y control de inventarios para mejorar los procesos de estas actividades de la tienda InVirtual Store.

### ***1.4.2 Objetivos Específicos***

- Analizar las actividades y procesos que realizan los administradores, empleados y clientes en la tienda.
- Construir un sistema web que permita la administración y control de inventario, pedidos y pagos pendientes.
- Implementar el sistema en un alojamiento web realizando pruebas funcionales para verificar el correcto desempeño de la aplicación.

- Evaluar el rendimiento del sistema mediante métricas que determinen el desempeño y calidad de este.

## **1.5 METODOLOGÍA**

Para el desarrollo del proyecto se considera utilizar la metodología SCRUM que proporciona un enfoque iterativo paso a paso para ayudar a controlar el riesgo, crear valor agregado y adaptar soluciones a problemas complejos.

El objetivo de este método de trabajo es obtener los mejores resultados posibles, el cual incluye una idea del dominio empírico de procesos. Por lo cual Scrum utiliza el avance continuo del proyecto para la planificación y lanzamiento del sistema. En esta metodología, el proyecto se divide en períodos de labor cortos llamados sprints. Suelen durar de una a tres semanas. Al final de cada iteración, los miembros se reúnen para estimar los avances del proyecto y planificar los próximos pasos. Esto permite corregir o reorientar el rumbo del proyecto una vez finalizado el trabajo sin necesidad de suposiciones y proyecciones.

El equipo Scrum generalmente consta de grupos de trabajo de entre 3 a 9 personas, así como el Scrum Máster y el Product Owner. Cada rol tiene diferentes responsabilidades y deben administrarse de diferentes maneras entre sí y para el resto de la organización. (Sordo Ana, 2021)

## **2. ANÁLISIS Y DISEÑO**

En este capítulo se analizará los problemas operacionales que presenta el emprendimiento InVirtual Store, además de la creación del diseño del sistema con los diferentes modelos y diagramas que serán utilizados para la construcción del sistema.

### **2.1 ANÁLISIS DEL PROBLEMA**

El siguiente subcapítulo se centra en el análisis de la problemática sobre el caso de estudio InVirtual Store cubriendo su historia, proceso de venta, administración de inventario, revisión de los pedidos. Además de identificar los requerimientos de la tienda.

#### **2.1.1 *InVirtual Store***

Es un emprendimiento creado por Carolina Larco y Daniel Ripalda, cuya actividad principal es la importación de productos desde Estados Unidos para comercializarlos en Ecuador. Como medio de venta utilizan las redes sociales. La idea empezó cuando Carolina en una visita a familiares que residen en la ciudad de Gaithersburg, Maryland; notó una oportunidad de negocio al observar la diferencia abismal de precios que existen entre las tiendas estadounidenses y sus similares en Ecuador con precios que muchas veces duplican o triplican su valor.

En noviembre del 2017, al retorno de su viaje a Quito, trajo consigo varias prendas de vestir de marcas reconocidas como: Adidas, Nike, Puma, Calvin Klein, Victoria Secret, entre otras; con el objetivo de venderlo a un precio menor al establecido por las tiendas oficiales de estas marcas en el país.

Hoy en día, para la importación de los productos, la tienda tiene rentado un pequeño depósito en la ciudad de Miami, donde realizan pedidos a través Amazon, o directamente a las páginas específicas de ciertas marcas. Posteriormente recurren al Servicio de Courier “Expresito”, que se

encarga de recoger los paquetes del depósito y hacer el proceso de importación por vía aérea o marítima hasta la ciudad de Guayaquil. Finalmente, los paquetes llegan a Quito por transporte terrestre.

Desde que empezó emprendimiento, han importado prendas de vestir de todo tipo, relojes, perfumes, bolsos, vitaminas y artículos tecnológicos pequeños como focos o altavoces inteligentes. Su nicho de mercado es principalmente niños y mujeres, por lo que la mayoría de sus productos están destinados a estos grupos. Además, aceptan pedidos personalizados siempre y cuando el nivel de dificultad de importación sea moderado.

En sus inicios para conseguir ventas utilizaban la publicidad de boca a boca. Posteriormente optaron por aprovechar la funcionalidad de los estados de WhatsApp como medio para promocionar sus productos a todos sus contactos a través de fotos. Actualmente dicha técnica se expandió con la creación de páginas de Facebook e Instagram de la tienda.

### ***2.1.2 Cadena de Suministro***

Para el desarrollo de cada negocio sin importar su tamaño, la cadena de suministro es parte fundamental para el negocio, con el objetivo de generar un mayor impacto para sus clientes actuales y así como para sus clientes potenciales.

La cadena de suministros es una conjunción de las unidades del negocio multi relacional, que combina toda la integración administrativa. Está sustentada como una conjunción de procesos que proponen una forma efectiva de control y administración de las transacciones comerciales y relaciones entre las diversas unidades de negocio. (Chase & Jacobs, 2011)

Dentro de los preceptos de la cadena de suministro, el que más resalta es conocer que este proceso busca contar con un panorama completo del suministro. Además, muestra un análisis de las actividades que realizan en conjunto los clientes y vendedores. Se enfoca en los aspectos que aportan valor comercial, tales como ofrecer una mayor calidad, cumplimiento de los pedidos, mayor velocidad para introducir nuevos productos y tecnologías que aporten a cada uno de estos aspectos. (Anne Millen Porter, 1997)

Está compuesta de 4 etapas, 3 de las cuales tienen una relación directa con las actividades realizadas por InVirtual Store, que son:

- **Abastecimiento o suministro:** Se concentra en el cómo, donde y cuando se consiguen los materiales tanto para la venta o fabricación de algún producto. Esta etapa se centra en las actividades de compra y abastecimiento de las provisiones necesarias para el desarrollo de las funciones a las que se dedica la empresa. (Giannice, 2009)
  
- **Distribución:** Se concentra que los productos terminados lleguen al cliente final mediante cualquier medio de distribución que dispongan como almacenes o distribución en línea. El producto es transportado hasta su destino final, según el acuerdo que se llega entre el vendedor y el consumidor, estableciendo la locación de entrega y el transporte en que llegará. (Díaz et al., 2012)
  
- **Consumidor:** Este apartado es tanto una etapa, como el actor más importante de la cadena de suministro. Es una entidad que puede ser persona u organización, que busca adquirir los



bienes o servicios que brinda el vendedor. Cuenta con una serie de necesidades, además cuenta con la disponibilidad de los medios económicos necesarios para satisfacer sus necesidades a través del mercado. (Ramírez Sánchez et al., 2022)

### **2.1.3 *Proceso del Negocio***

El negocio está centrado en el manejo de inventario de productos importados, registro de ventas, y control de pagos realizados por los clientes. Además, el proceso se divide entre tres tipos distintos de usuarios: administrador, empleados (vendedores y reparadores), y clientes de la tienda.

A continuación, se muestran cada uno de los procesos que componen las operaciones de la tienda InVirtual Store, mismos que son complementarios y se van ampliando en cada literal.

#### **2.1.3.1. *Registro de inventario***

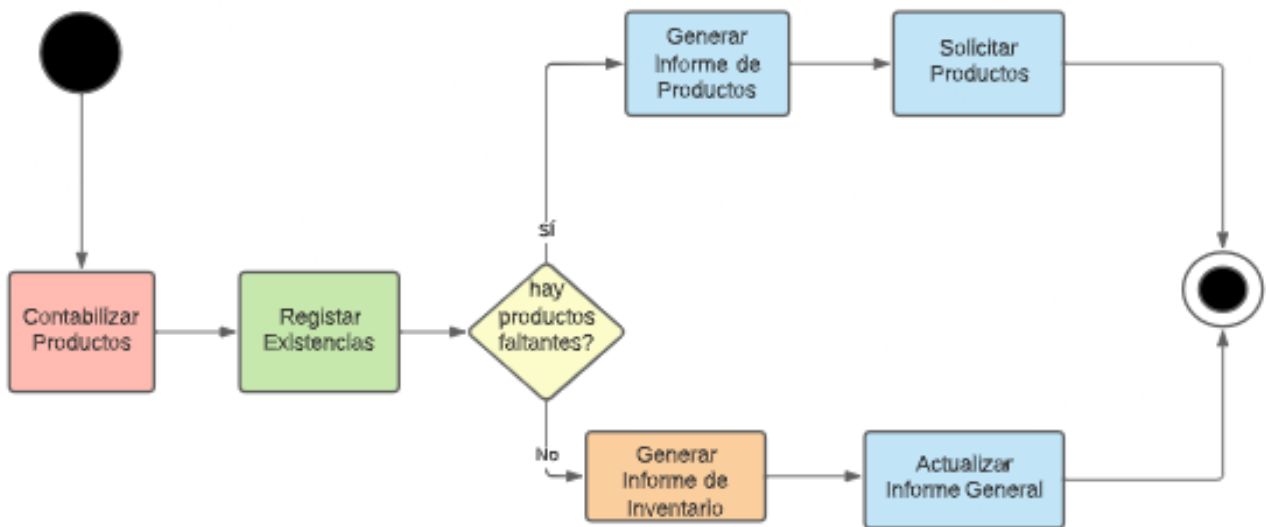
Este proceso es ejecutado tanto por el administrador como por el personal vendedor de la tienda, mismo que se realiza cada vez que llegan nuevos productos importados.

El proceso inicia cuando, separan, organizan y contabilizan la mercadería, con el objetivo de catalogar y tener registro de las existencias de cada producto.

En el caso de productos nuevos se genera un nuevo registro en la de hoja de cálculo destinada a tener inventario de estos con su información general, y respectivo número de existencias en un formato establecido tal y como se muestra en la Figura 2.

Por otro lado, si se encuentran productos agotados o que cuentan con pocas unidades, se procede a guardar en un registro distinto a estos productos, para que sean solicitados para su importación en el siguiente pedido de mercadería.

**Figura 1**  
*Diagrama de flujo del proceso de registro de inventario*



*Nota: Proceso de registro de existencias del inventario. Elaborado por: Los autores.*

## Figura 2

### Catálogo de Productos InVirtual Store

Productos Catálogo InVirtualStore									
CODIGO	Proveedor	Producto	Datos del producto	Detalles	Precio Mercado	Precio Proveedor	Existencias	Variacion1	Variación 2
RTE001	Ruta tu estilo	Camiseta deportiva	Esta es una camiseta de poliester de maxima reistencia, con tecnologia dryfit, ideal para hacer deporte de alta competencia.	Material: Poliester Peso: En 2 gr Tamaño: 30 Cm Origen: Estados Unidos	\$50,00	\$45,00	10	Colores: Rojo Banco Negro Azul	Talla: S M L
RTE002									
RTE003									
RTE004									
RTE005									
RTE006									
RTE007									

*Nota: Hoja de cálculo utilizada para el registro de inventarios. Elaborado por: InVirtual Store.*

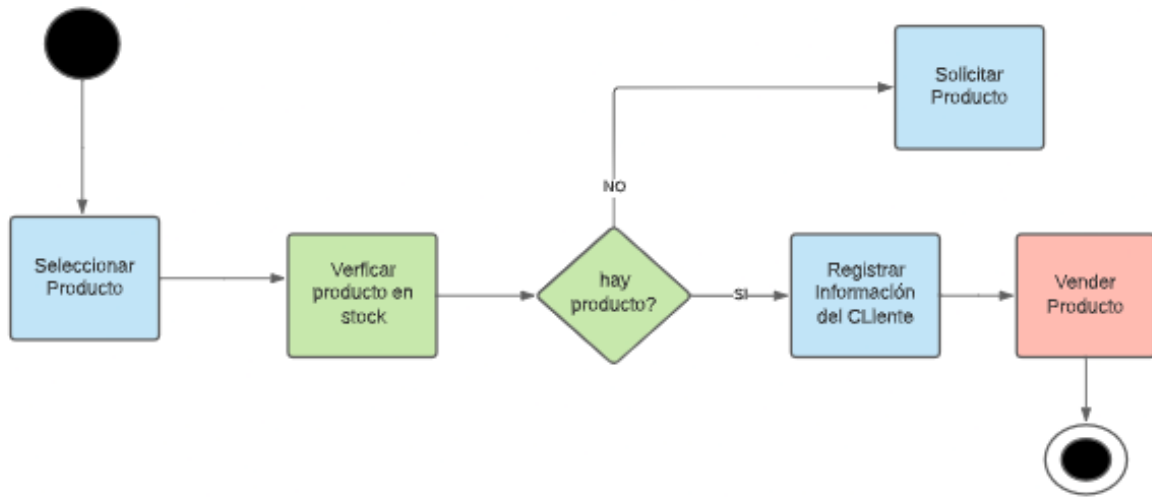
#### 2.1.3.2. Realización de pedido

Dentro de este proceso participa tanto la persona que registra la venta, que puede ser el administrador de la tienda, o un empleado vendedor; así como el cliente quien realiza un pedido.

Este proceso inicia cuando el cliente mira el producto que desea comprar, por cualquiera de los medios de venta (redes sociales), y procede a contactar con la persona encargada de la venta mediante llamada telefónica o mensaje de texto.

El vendedor verifica que el producto o productos solicitados se encuentren disponibles, si lo están, el proceso continúa, donde el empleado guarda la información del cliente y procede a registrar la venta realizada; caso contrario se informa al cliente de dicha novedad y se propone la opción de poder completar su pedido, solicitándolo para la siguiente importación de productos.

**Figura 3**  
*Diagrama de Flujo realización de un pedido*



*Nota: Proceso de realización de un pedido. Elaborado por: Los autores.*

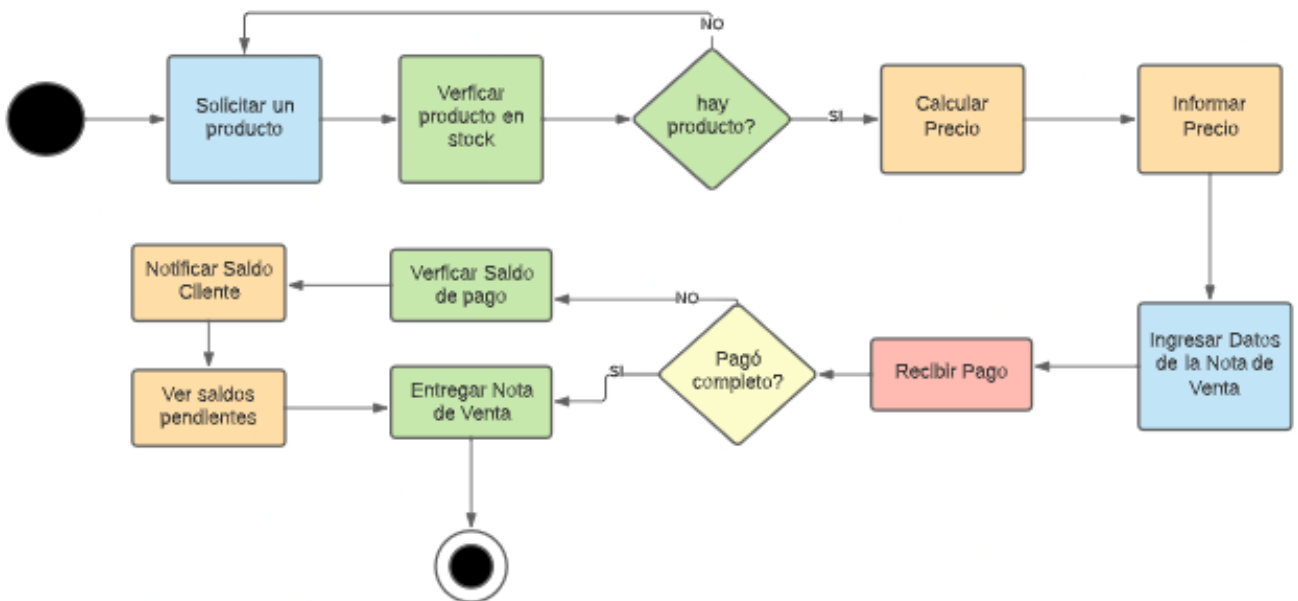
### **2.1.3.3. Venta del producto**

En este proceso participan tanto el personal vendedor, personal repartidor, y comprador. En primer lugar, el comprador solicita su pedido, donde la persona encargada de la venta verifica su existencia en stock, esta actividad se muestra más a detalle en los procesos previos.

Si uno de los productos no se encuentra en stock el cliente puede escoger otro producto. Caso contrario la venta prosigue donde el personal vendedor calcula su precio dependiendo del peso y el lugar de entrega del producto y procede a informar al cliente; una vez aceptado el pedido por el cliente, el empleado procede a generar una nota de venta.

A continuación, un repartidor se encarga de entregar el producto, donde recibe el pago del pedido, y procede a verificar si el monto cancelado corresponde al valor total del pedido. Si el valor es correcto, culmina el pedido entregando una nota de venta que da constancia del pago. Caso contrario, se verifica el saldo pendiente notificando tanto al cliente como al administrador el valor restante por pagar del pedido.

**Figura 4**  
*Diagrama de Flujo de la venta del producto*



*Nota: Proceso de la venta de un producto. Elaborado por: Los autores.*

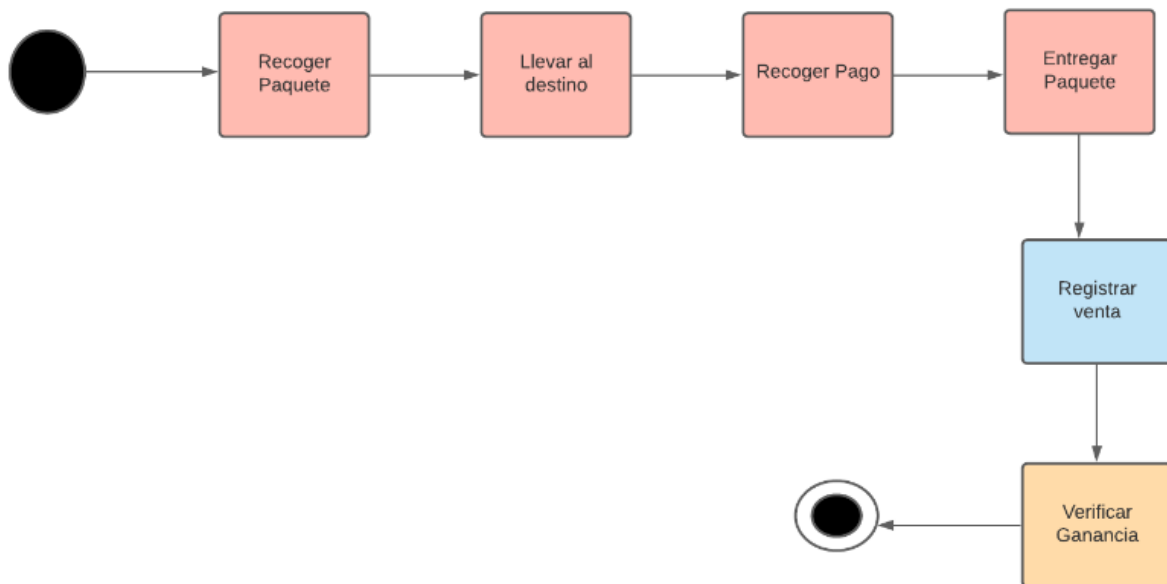
#### 2.1.3.4. *Repartición del Pedido*

Este proceso está destinado de forma exclusiva al personal repartidor, donde este encarga de recoger el paquete (pedido), obtendrá la información del cliente que fue recopilada por el vendedor y procede a llevarlo a su destino final.

Si la entrega es dentro de la ciudad de Quito, llegado al destino, recoge el pago realizado por el cliente, entrega el paquete, y registra la venta. Si el destino final es en otra provincia, se lleva el paquete al Terminal Terrestre y entrega al respectivo servicio de transporte interprovincial de acuerdo con el lugar de entrega, y toma registro del número del paquete enviado.

Finalmente se realizan las verificaciones explicadas en el proceso anterior. Una vez culminada la entrega, el repartidor obtendrá su ganancia acordada previamente con el administrador, dependiendo del sitio de entrega y peso del paquete.

**Figura 5**  
*Diagrama de Flujo de la repartición del pedido*



*Nota: Proceso de la repartición de un pedido. Elaborado por: Los autores.*

#### **2.1.4 Flujo de Datos del Proceso**

Las actividades centrales de la tienda se basan en la administración de inventario y el control de ventas, donde cada usuario interactúa con este proceso con varios datos de entrada y salida, mismo que son importante reconocerlos para lograr un entendimiento más completo del funcionamiento del proceso.

A continuación, dentro de este análisis se recurre al Diagrama de Contexto. Este diagrama, también conocido como flujo de datos Nivel 0, es un medio para definir cada una de las entidades participantes en los procesos principales de la tienda. Establece límites en el alcance, las partes interesadas que intervienen en el desarrollo de la aplicación, así como su relación e interacción con los componentes externos (entradas y salidas) que influyen en el sistema. (Pedriquez Daleska, 2022)

Tal y como se muestra en la figura 6 en la parte central se encuentra el sistema que se quiere definir, con sus respectivos procesos y responsabilidades del trabajo.

A los lados se plasmaron las entidades externas o agentes, donde se identificaron cuatro de ellos que intervienen en el transcurso de todo el proceso: Administrador, Cliente, Repartidor y Empleado.

Para la comunicación entre las entidades y la parte central, se establecen las líneas de flujo, que muestran los datos que fluyen entre estos dos aspectos, o las formas específicas en que las entidades interactúan con el producto.

En el lado izquierdo se muestra las entradas que cada entidad brinda al sistema según cada proceso en el que se ve involucrado, especificando las actividades que realiza cada uno, tal y como se lo exploró en el subcapítulo anterior.

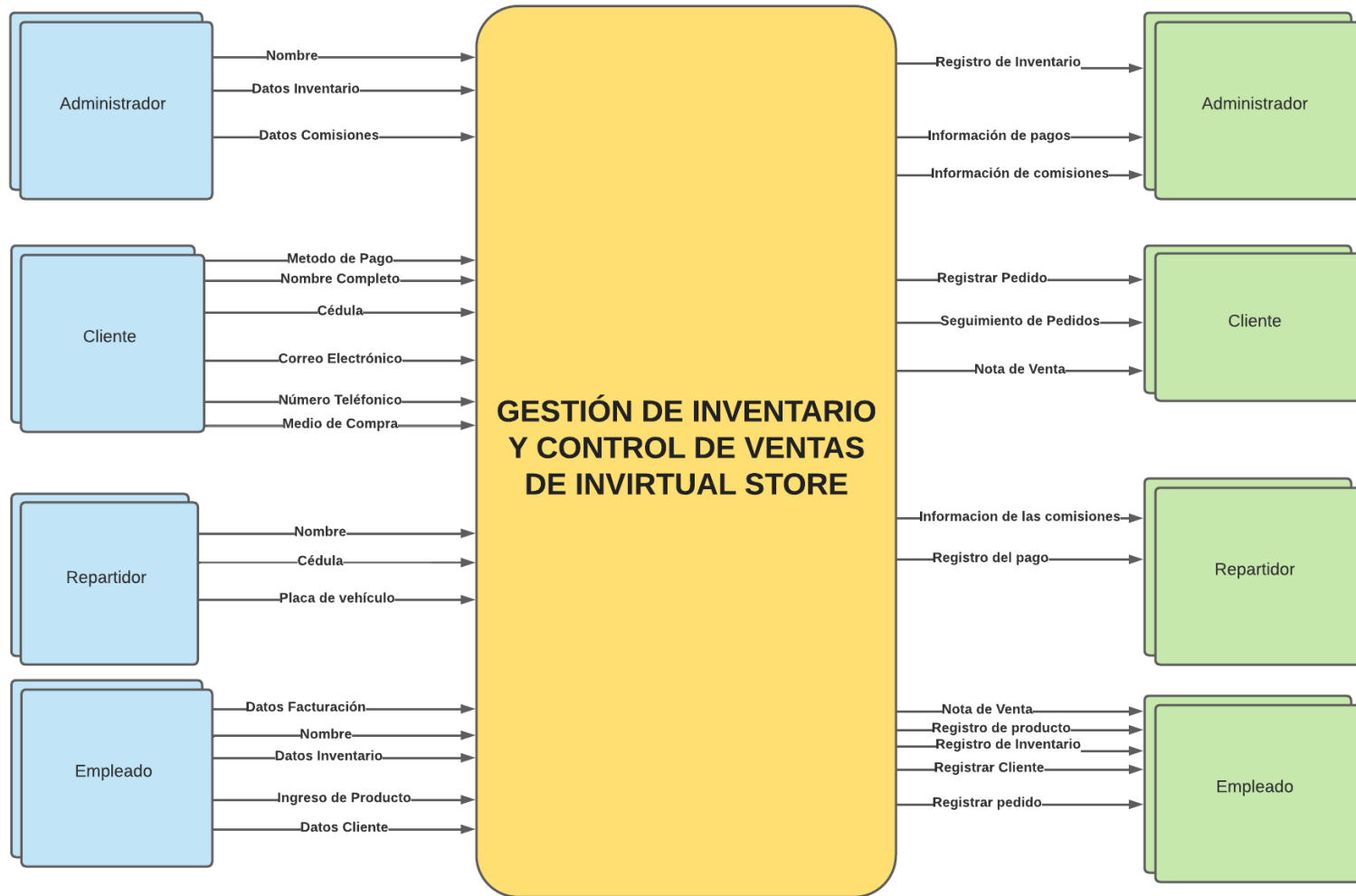
Mientras en el lado derecho se muestra las salidas que da el proceso central y son receptadas por cada una de las entidades, de acuerdo con los flujos establecidos como entradas del sistema, en

conjunto los resultados que se generan en el transcurso de cada uno de los procesos de la operación de la empresa.



**Figura 6**

*Diagrama de Contexto de Gestión de inventario y Control de ventas*



*Nota: Procesos de Gestión de inventario y Control de ventas. Elaborado por: Los autores.*

## **2.2 ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE**

El presente subcapítulo explicará los requerimientos de software que se han realizado de una forma detallada para cumplir con los objetivos del proyecto del caso de estudio InVirtual Store; asimismo se indicará los procesos, características y funcionalidades que contiene el sistema.

### ***2.2.1 Perspectiva del Producto***

El sistema de administración de ventas y control de inventarios será una aplicación web, que funciona como un remplazo de los procesos de registro de inventario realizado de forma manual o mediante el uso de hojas de cálculo de Microsoft Excel. Para el control de ventas en la tienda, el sistema reemplazará al método de registro de deudas tanto para la administración de la tienda, como para el cliente, que era realizado netamente de forma manual.

El sistema contará con un Servidor de Aplicaciones, y una Base de Datos alojado en un servicio en la nube, haciendo que el sistema sea accesible en cualquier lugar, y con disponibilidad inmediata de los datos de los productos y de cada uno de los clientes.

### ***2.2.2 Funcionalidades del Producto***

El sistema será desarrollado como un aplicativo web, los empleados y clientes podrán acceder mediante un inicio de sesión con el cual se autenticarán y se les permitirá el acceso a los siguientes módulos dependiendo los permisos que tenga el usuario.

- **Administración:** En este módulo se administra los todos los registros de productos, empleados, clientes y repartidores, también se gestiona la repartición y pagos de los pedidos realizados.
- **Pedidos:** En este módulo se podrá dar seguimiento a los pedidos por entregar y asignados a cada repartidor de la tienda, al igual que ver el estado del pedido y registrar los pagos, por último, mostrar la nota de venta al cliente.

### 2.2.3 Ambiente de Operaciones

El software se ejecutará en un servidor bajo el Sistema Operativo Linux, en conjunto con un servidor Flexible en la nube para su base de datos, con el objetivo de ser accesible a través de la web.

Para el correcto funcionamiento de la aplicación es necesario tener cubiertos una serie de requisitos, específicamente en software. Al ser una aplicación web, para poder ejecutar la aplicación será necesario contar con navegadores web que están disponibles mediante una PC o cualquier dispositivo móvil.

Debido a las funcionalidades con las que cuenta la aplicación, para un correcto y estable funcionamiento será necesario contar con navegadores como:

- Google Chrome 81.0.x o superior
- Microsoft Edge 81 o superior
- Mozilla Firefox 2.0 o superior
- Otros navegadores basados en Chromium 81.0.x (Opera - Brave).

## 2.2.4 Características de cada perfil de usuario

**Tabla 1**

*Perfil de usuario Administrador*

Perfil de usuario	Administrador
Experiencia	Este usuario debe tener un amplio conocimiento de los procesos que se manejan en el sistema.
Destreza	Gestionar y repartir los pedidos
Actividades	Registra y gestiona la actualización de los productos, acceder a todos los registros guardados de la base de datos, dar seguimiento a los pedidos asignados, por entregar y pendientes de pago.

*Nota: Actividades del usuario con perfil Administrador. Elaborado por: Los autores.*

**Tabla 2**

*Perfil de usuario Empleado*

Perfil de usuario	Empleado
Experiencia	Uso de páginas de administración de inventario.
Destreza	Gestionar y registrar los productos, clientes y proveedores.
Actividades	Registra y gestiona la actualización de los productos, clientes, proveedores.

*Nota: Actividades del usuario con perfil Empleado. Elaborado por: Los autores.*

**Tabla 3**

*Perfil de usuario Repartidor*

Perfil de usuario	Repartidor
Experiencia	Este usuario se encarga de repartir los pedidos a los clientes.
Destreza	Entrega de pedidos a clientes
Actividades	Iniciar los pedidos asignados y registra los pagos cuando se entregue el pedido al cliente y ver sus ganancias.

*Nota: Actividades del usuario con perfil Repartidor. Elaborado por: Los autores.*

**Tabla 4**

*Perfil de usuario Cliente*

Perfil de usuario	Cliente
Experiencia	No necesario
Destreza	Utilizar páginas web
Actividades	Dar seguimiento y ver el detalle a sus pedidos, mostrar los pagos pendientes y por último revisar sus compras con visualizaciones de notas de ventas.

*Nota: Actividades del usuario con perfil Cliente. Elaborado por: Los autores.*

### **2.2.5 Restricciones**

En esta subsección se describirá algunas de las limitaciones que se imponen al momento del desarrollo del sistema, al igual que cuando este sistema se implemente en la nube.

### **2.2.6 Políticas**

La aplicación se desarrollará utilizando un software de código abierto, por lo que no se tendrá que pagar licencias por utilizar. El motor de base de datos será PostgreSQL, contará con una arquitectura MVC, para el lenguaje de programación se utilizará Javascript con el marco de trabajo Node.js y Handlebars.js. El servidor web de aplicaciones que se utilizará es Azure App Services Environment v3.

### **2.2.7 Limitaciones de Hardware**

Para el despliegue de la aplicación se necesitará un servicio alojado en la nube en el cual se subirá y configurará la base de datos PostgreSQL, además de un servidor web Linux que ejecute la aplicación.

El sistema debe ser instalado en un servidor de aplicaciones que sea compatible con el motor de base de datos utilizado, así mismo mediante un navegador web el sistema deberá tener una buena conexión a internet para que la aplicación funcione correctamente.

### **2.2.8 Requisitos futuros**

Cada sistema se considera escalable si puede adaptarse a mayores requisitos de procesamiento de datos y un aumento de funciones adicionales, de modo que el sistema podrá ampliar su funcionalidad con repartición automática de los pedidos, trackeo en tiempo real de los estados de los pedidos, conexión a módulos nuevos que requiera el sistema, informes estadísticos de ventas y compras, entre otros.

### 2.2.9 *Requerimientos funcionales específicos*

A continuación, se detallan todos los requerimientos funcionales que debe cumplir el sistema. Todos estos requerimientos establecidos en este documento son importantes, cada uno de estos requisitos especificados aquí describen el comportamiento del sistema teniendo en cuenta el criterio del usuario.

**Tabla 5**  
*Requerimiento 01 Registro de Inventario*

Número de requisito	RF01
Nombre de requisito	Registro de Inventario
Dependencias	No depende de otros requisitos
Entrada	Nombre, Proveedor, imagen, categoria, color, precio mercado, precio proveedor, cantidad, origen, peso en libras, talla
Proceso	Ingresar el producto
Salida	Información del producto guardada.
Prioridad	Muy Alta
Comentarios	Ayuda digital para el control de inventario.

*Nota: Requisito de software RF01, entradas, salidas, proceso y prioridad del requerimiento. Elaborado por: Los autores.*

**Tabla 6**  
*Requerimiento 02 Registro del Cliente*

Número de requisito	RF02
Nombre de requisito	Registro del Cliente
Dependencias	No depende de otros requisitos.
Entrada	Nombre y apellido, cédula, correo electrónico, edad, teléfono, medio de compra, género
Proceso	Ingresar el cliente
Salida	Información del cliente guardado.
Prioridad	Alta
Comentarios	Necesario para que el cliente tenga una información completa de los pedidos que han realizado

*Nota: Requisito de software RF02, entradas, salidas, proceso y prioridad del requerimiento. Elaborado por: Los autores.*

**Tabla 7***Requerimiento 03 Registro de Pedido*

Número de requisito	RF03
Nombre de requisito	Registro de Pedido
Dependencias	Dependencia absoluta del requerimiento 01 y 02
Entrada	Cédula del cliente,
Proceso	Ingresar productos y cantidad, registrar el pedido
Salida	Información del pedido grabada.
Prioridad	Muy alta
Comentarios	Principal aspecto del programa, para que el administrador y los empleados tenga una visualización completa de los productos por vender.

*Nota: Requisito de software RF03, entradas, salidas, proceso y prioridad del requerimiento. Elaborado por: Los autores.*

**Tabla 8***Requerimiento 04 Seguimiento del pedido para el administrador*

Número de requisito	RF04
Nombre de requisito	Seguimiento del pedido para el administrador
Dependencias	Depende del requerimiento funcional 03
Entrada	
Proceso	Consulta de los pedidos con sus estados.
Salida	Visualización de todos los pedidos realizados
Prioridad	Muy alta
Comentarios	Importante para que el administrador tenga el conocimiento del estado de los pedidos.

*Nota: Requisito de software RF04, entradas, salidas, proceso y prioridad del requerimiento. Elaborado por: Los autores.*

**Tabla 9***Requerimiento 05 Seguimiento del pedido para el cliente*

Número de requisito	RF05
Nombre de requisito	Seguimiento del pedido para el cliente
Dependencias	Depende del requerimiento funcional 03
Entrada	
Proceso	Consulta de los pedidos del cliente con sus estados.
Salida	Visualización de los pedidos realizados
Prioridad	Alta
Comentarios	Importante debido a que esto es el plus del programa evitando que los usuarios gasten más de lo establecido.

*Nota: Requisito de software RF05, entradas, salidas, proceso y prioridad del requerimiento. Elaborado por: Los autores.*

**Tabla 10***Requerimiento 06 Registro de pagos*

Número de requisito	RF06
Nombre de requisito	Registro de pagos
Dependencias	Depende del requerimiento funcional 03
Entrada	Cantidad por pagar, método de pago, tipo de pago
Proceso	Guardar el pago
Salida	Información de pago guardada
Prioridad	Muy Alta
Comentarios	Bajo este requerimiento se podrá hacer seguimiento de los pagos que realicen los clientes y hacer un seguimiento a saldos pendientes.

*Nota: Requisito de software RF06, entradas, salidas, proceso y prioridad del requerimiento.  
Elaborado por: Los autores.*

**Tabla 11***Requerimiento 07 Visualización y pagos pendientes*

Número de requisito	RF07
Nombre de requisito	Visualización y pagos pendientes
Dependencias	Depende del requerimiento funcional 06
Entrada	
Proceso	Consulta de los pagos realizados o por pagar
Salida	Revisión del estado de los pagos.
Prioridad	Alta
Comentarios	Permitirá tener un mayor control en los cobros de los pagos, evitando así pérdidas a la empresa por pedidos sin cobrar que antes dependían netamente de la memoria.

*Nota: Requisito de software RF07, entradas, salidas, proceso y prioridad del requerimiento.  
Elaborado por: Los autores.*

**Tabla 12***Requerimiento 08 Notas de ventas*

Número de requisito	RF08
Nombre de requisito	Notas de ventas
Dependencias	Depende del requerimiento funcional 06
Entrada	Datos de la nota de venta
Proceso	Realización de cálculo del IVA
Salida	Nota de venta registrada
Prioridad	Muy Alta
Comentarios	Necesario para los cálculos finales que realizará el programa y registrar de forma legal las ventas realizadas.



*Nota: Requisito de software RF08, entradas, salidas, proceso y prioridad del requerimiento.  
Elaborado por: Los autores.*

**Tabla 13**

*Requerimiento 09 Visualización de las comisiones de los repartidores*

Número de requisito	RF09
Nombre de requisito	Visualización de las comisiones de los repartidores
Dependencias	Depende del requerimiento 06
Entrada	
Proceso	Consulta de los montos de las comisiones
Salida	Vista de las comisiones obtenidas por el repartidor
Prioridad	Muy Alta
Comentarios	Tanto el administrador como los empleados repartidores podrán revisar el monto de sus comisiones

*Nota: Requisito de software RF09, entradas, salidas, proceso y prioridad del requerimiento.  
Elaborado por: Los autores.*

### **2.2.10 Requerimientos no funcionales**

A continuación, se detallan todos los requerimientos no funcionales principales:

- **Interfaz del usuario**

El sistema debe ser adecuado e intuitivo para su uso, además de contener botones nombrados de acuerdo con sus acciones específicas, y mensajes que expliquen detalladamente las razones de un error que se pueda presentar para evitar equivocaciones en el sistema.

El sistema será responsive para que se adapte a cualquier tamaño de pantalla en la que se abra.

Utilizado para la facilidad y comunidad de los usuarios, logrando que ejecuten el software en cualquier tamaño de pantalla sin ningún problema.

- **Soporte**

El sistema contará con un servidor que se actualizará cada semana para no perder servicio constante, y tener información actualizada.

El sistema será web y debe soportar los navegadores Google Chrome, Mozilla Firefox, Microsoft Edge, Otros navegadores basados en Chromium.

- **Seguridad**

El acceso al sistema debe estar limitado por la contraseña asignada a cada usuario del sistema, Solo las personas registradas pueden iniciar sesión en el sistema Invirtual Store, estos usuarios podrán acceder a los distintos módulos de acuerdo con su perfil.

- **Disponibilidad**

Debido al manejo concurrente el sistema estará disponible las 24 horas del día ya que cuenta con un servidor web alojado en la nube.

## **2.3 DISEÑO DEL SISTEMA**

En este subcapítulo se detallan los diagramas y modelos utilizados en el sistema, así como la arquitectura que se ha considerado implementar para la solución de las distintas funcionalidades.

### **2.3.1 Arquitectura del Sistema**

En este apartado se revisará de forma detallada el patrón de arquitectura definido para la aplicación, así como la descripción de los componentes que contendrán.

#### **2.3.1.1. Modelo Vista Controlador (MVC)**

Es un patrón arquitectónico que divide la aplicación en tres partes lógicas: la parte del modelo, la vista y el controlador. Inicialmente era utilizado exclusivamente para interfaces gráficas de usuario de escritorio, pero hoy en día es utilizada para diseñar aplicaciones web o móviles.

Este patrón fue creado por Trygve Reenskaug en 1979. Su objetivo principal era resolver el problema que se generaba cuando los usuarios de los sistemas tienen que controlar un conjunto grande y complejo de datos.

MVC es un patrón arquitectónico, esto significa que gobierna toda la arquitectura de las aplicaciones. Esto quiere decir que no se centra en solventar problema técnico específico, sino que es utilizado para resolver problemas arquitectónicos, por lo que afecta a toda la arquitectura de la aplicación. (Singh, 2020)

Las razones principales por las que se usa este patrón arquitectónico son porque no permite redundancia en la estructura, y ayuda a crear una estructura sólida en las aplicaciones web. MVC cuenta con 3 componentes específicos:

- **Modelo:** Es el nivel más bajo, por lo tanto, es el responsable de mantener los datos, manejando los datos de forma lógica. El modelo es el que contiene la conexión a la base de datos, por lo que las actividades de lectura y escritura de datos son realizadas en este componente. Además, responde a las solicitudes del controlador, comunicándose con la base

de datos de un lado a otro, para luego proveer los datos necesarios al controlador. Finalmente cabe recalcar que el modelo nunca se comunica directamente con la vista.

- **Vista:** En este nivel, se realiza la representación de datos, es decir aquí se genera una interfaz de usuario (UI) para el sistema. En las aplicaciones web, al hablar del componente de vista, se refiere al apartado visual usualmente con componentes basados en HTML, CSS y JavaScript. Las vistas son creadas por los datos que recopila el componente del modelo, con datos que son procesados a través del controlador, por lo que la vista solo se comunica directamente con el controlador.
- **Controlador:** Este nivel es el principal de toda la arquitectura, debido a que es el componente que permite la interconexión entre las vistas y el modelo fungiendo como intermediario. El controlador no se preocupa por manejar la lógica de datos, simplemente le dice al modelo qué hacer. Después de recibir datos del modelo, los procesa y toma toda la información, enviándolo a la vista y dejando instrucciones sobre cómo representar la información al usuario. (Ahmad et al., 2022)

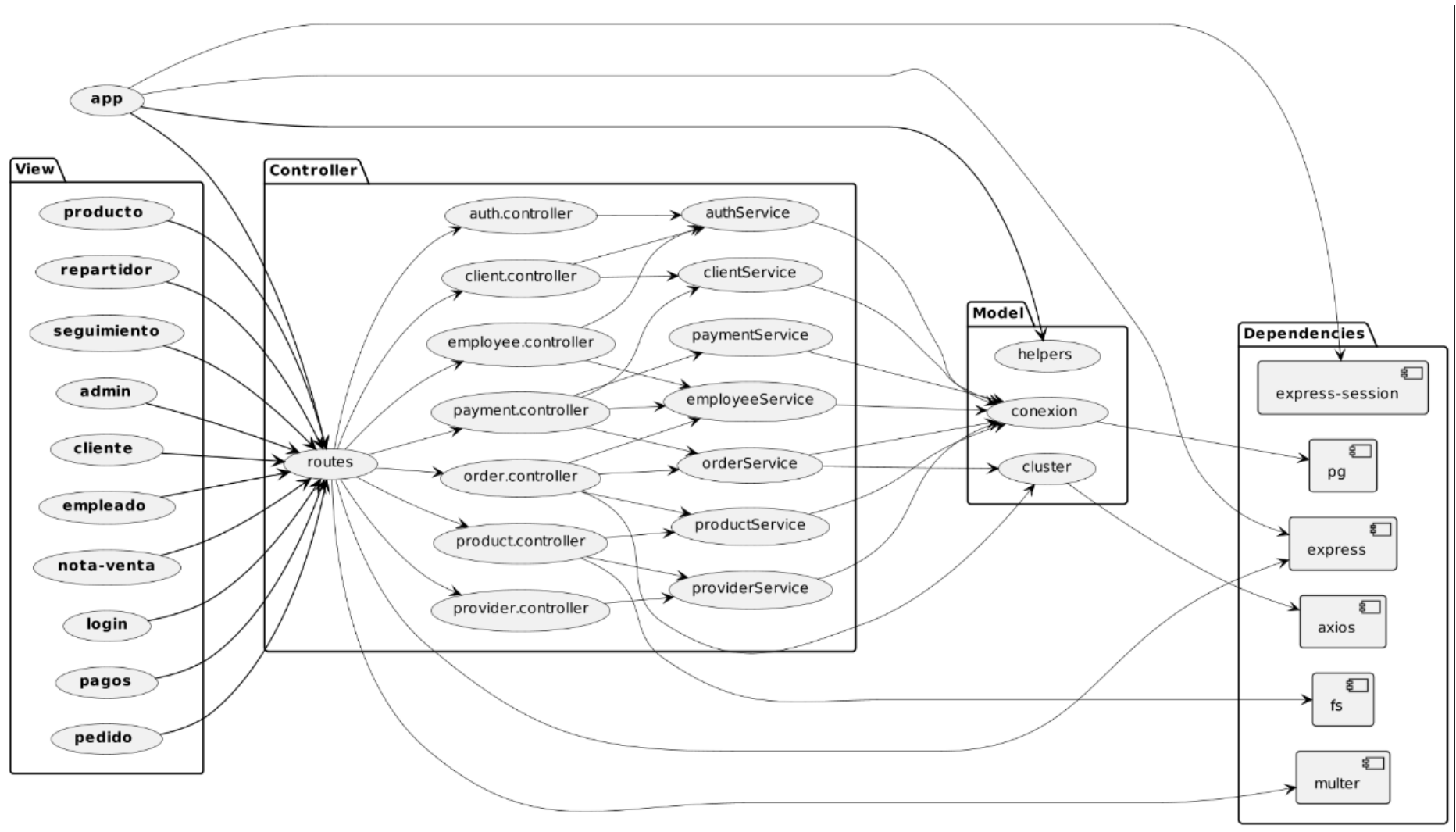
### **2.3.1.2. *Diagrama de Arquitectura***

A continuación, se muestra el diagrama de la arquitectura seguida, con sus respectivos directorios basados en el patrón de arquitectura MVC con los archivos necesarios para cumplir con su funcionalidad, además muestra cada dependencia que contendrá cada archivo tanto entre ellos, así como con dependencias de librerías externas propias de Node que ayudarán a ejecutar acciones centrales del software.

- El modelo será el encargado de manejar la conexión a la Base de Datos, y cada uno de los métodos HTTP que se utilizarán para las actividades de lectura y escritura de datos con la Base de Datos. Adicionalmente, posee funciones de ayuda (Helpers) que permitirán una comunicación más directa con las acciones del controlador.
- El controlador será el encargado de controlar las rutas del servidor, y los parámetros necesarios para la ejecución de los métodos HTTP establecidos en el modelo. El controlador está separado en controladores y servicios para cada funcionalidad implementada. Los servicios contendrán las llamadas a la Base de Datos. Mientras que los controladores tendrán las funciones de tratamiento de datos ejecutando la lógica de cada ruta.
- La vista contendrá una carpeta de parciales (Partials), que incluye a los elementos principales del diseño general de la página, mismos que son reutilizables a lo largo de la aplicación. Adicionalmente, contendrá otra subcarpeta con cada una de las páginas del sistema, desarrolladas bajo una base en HTML, en conjunto con las funcionalidades que ofrece la librería HandleBars.js.
- Adicionalmente a lo referido al Modelo, Vista, Controlador, se colocó dentro del diagrama una carpeta de componentes. Los componentes mostrados, son principales utilizados en el sistema, mismos que ayudan a manejar diversas acciones tales como ayudar con la conexión a la Base de Datos, manejo de archivos del sistema y creación de entorno del servidor son las que más destacan.

Las dependencias mencionadas, son representación de las librerías que ciertas funciones del controlador tienen acceso a ellas para usar sus funciones internas para lograr un rápido y sencillo procesamiento de acciones que son requeridas para la aplicación.

**Figura 7**  
*Arquitectura del Sistema*

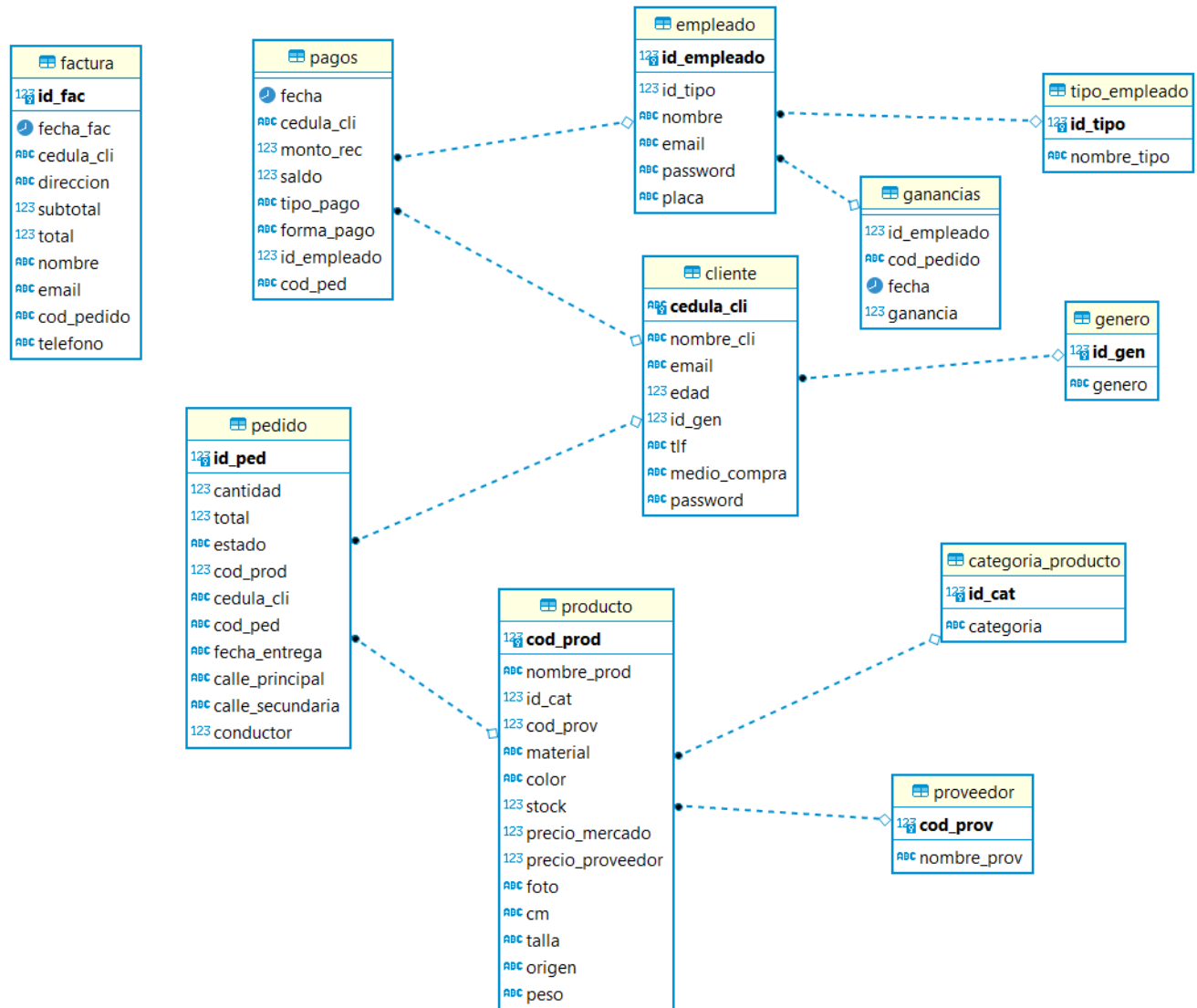


*Nota: Diagrama de arquitectura del sistema. Elaborado por: Los autores.*

### 2.3.2 Base de datos

En esta sección se describe el diseño conceptual de la base de datos (BBDD) al igual que las relaciones de las tablas y su orientación de los datos.

**Figura 8**  
*Diagrama Relacional de la Base de Datos*



*Nota: Diagrama Relacional de la Base de Datos. Elaborado por: Los autores.*

En esta parte se explicará cada una de las tablas, claves primarias, claves foráneas y los tipos de datos que contiene la base de datos, en la cual se almacenará toda la información de la tienda InVirtual Store.

**Tabla producto:** En esta tabla se almacena todos los productos que el administrador o empleado registre dentro del sistema.

**Tabla 14**  
*Tabla Producto*

Campo	PK	FK	Tipo	Descripción
cod_prod	•		Integer	Código de identificación del producto
nombre_prod			Varchar(100)	Nombre del producto
id_cat		•	Integer	Código de identificación de la categoría del producto
cod_prov		•	Integer	Código de identificación del proveedor del producto
material			Varchar(50)	Material del producto
color			Varchar(50)	Color del producto
stock			Integer	Cantidad del producto
precio_mercado			Float	Precio de mercado del producto
precio_proveedor			Float	Precio de proveedor del producto
foto			Varchar(300)	Imagen visual del producto
talla			Varchar(5)	
peso			Float	Peso del producto

*Nota: Tabla Producto de la Base de Datos. Elaborado por: Los autores.*

**Tabla cliente:** Guarda los datos desde el formulario registrar cliente.

**Tabla 15**  
*Tabla cliente*

Campo	PK	FK	Tipo	Descripción
cedula_cli	•		Varchar(10)	Número de identificación del cliente
nombre_cli			Varchar(200)	Nombre del cliente
email			Varchar(200)	Correo electrónico del cliente
edad			Integer	Edad del cliente
id_gen		•	Float	Id del género del cliente
tlf			Varchar(10)	Número de teléfono del cliente
medio_compra			Varchar(100)	Medio por donde compra el cliente



				WhatsApp, Facebook, Messenger
password			Varchar(10)	Contraseña del cliente

*Nota: Tabla cliente de la Base de Datos. Elaborado por: Los autores.*

**Tabla pedido:** Almacena la información de los pedidos registrados con el tipo de estado

**Tabla 16**

*Tabla pedido*

Campo	PK	FK	Tipo	Descripción
id_ped	•		Integer	Id del pedido
cantidad			Integer	Cantidad
total			Float	Total, del pedido realizado.
estado			Varchar(50)	Estado del pedido: Entregado-Cancelado Conductor Asignado Conductor en Camino Entregado-Pago Parcial Pedido Registrado
cod_prod		•	Integer	Código del producto
cedula_cli		•	Varchar(100)	Número de identificación del cliente
cod_ped			Varchar(20)	Código del pedido
fecha_entrega			Date	Fecha de entrega del pedido
calle_principal			Varchar(200)	Calle principal del pedido
calle_secundaria			Varchar(200)	Calle secundaria del pedido
conductor			Integer	Id del conductor del pedido

*Nota: Tabla pedido de la Base de Datos. Elaborado por: Los autores.*

**Tabla pagos:** Almacena los pagos de los clientes

**Tabla 17**

*Tabla pagos*

Campo	PK	FK	Tipo	Descripción
fecha			Date	Fecha de pago del pedido
cedula_cli	•		Varchar(10)	Número de identificación del cliente
monto_rec			Float	
saldo			Float	
tipo_pago			Varchar(50)	Tipo de pago
forma_pago			Varchar(100)	Forma de pago
id_empleado		•	Integer	Código del empleado
cod_ped		•	Integer	Código del pedido

*Nota: Tabla pagos de la Base de Datos. Elaborado por: Los autores.*

**Tabla empleado:** Guarda los datos desde el formulario de registro de empleados

**Tabla 18***Tabla empleado*

Campo	PK	FK	Tipo	Descripción
id_empleado	•		Integer	Id del empleado
id_tipo		•	Integer	Id del tipo de empleado
nombre			Varchar(200)	Nombre del empleado
email			Varchar(50)	Correo electrónico del empleado
password			Varchar(50)	Contraseña del empleado
placa			Varchar(10)	Placa si es un empleado repartidor

*Nota: Tabla empleado de la Base de Datos. Elaborado por: Los autores.***Tabla ganancias:** Almacena las ganancias del empleado con perfil repartidor**Tabla 19***Tabla ganancias*

Campo	PK	FK	Tipo	Descripción
id_empleado	•		Integer	Id del empleado
codigo_pedido		•	Integer	Código del pedido
fecha			Date	Fecha
ganancia			Double	Número de ganancia del empleado repartidor

*Nota: Tabla ganancias de la Base de Datos. Elaborado por: Los autores.***Tabla proveedor:** Almacena los datos del proveedor desde el formulario registro del proveedor**Tabla 20***Tabla proveedor*

Campo	PK	FK	Tipo	Descripción
cod_prov	•		Varchar(10)	Código del proveedor
Nombre_prov			Varchar(200)	Nombre del proveedor

*Nota: Tabla proveedor de la Base de Datos. Elaborado por: Los autores.***Tabla categoria\_producto:** Almacena el tipo de categoría del producto desde el formulario de registro**Tabla 21***Tabla categoria\_producto*

Campo	PK	FK	Tipo	Descripción
id_cat	•		Integer	Id de la categoría
categoria			Varchar(200)	Nombre de la categoría

				Ropa, Cosméticos, Perfumes, Salud
--	--	--	--	-----------------------------------

*Nota: Tabla categoria\_producto de la Base de Datos. Elaborado por: Los autores.*

**Tabla genero:** Muestra en el formulario los géneros disponibles el en formulario

### Tabla 22

*Tabla genero*

Campo	PK	FK	Tipo	Descripción
id_gen	•		Float	Id del género
genero			Varchar(50)	Nombre del género Masculino, Femenino, No definido

*Nota: Tabla genero de la Base de Datos. Elaborado por: Los autores.*

**Tabla tipo\_empleado:** Muestra los tipos de empleados

### Tabla 23

*Tabla tipo\_empleado*

Campo	PK	FK	Tipo	Descripción
id_tipo	•		Integer	Id del tipo de empleado
nombre_tipo			Varchar	Nombre del tipo de empleado Administrador, Empleado, Repartidor

*Nota: Tabla tipo\_empleado de la Base de Datos. Elaborado por: Los autores.*

**Tabla factura:** Detalla los datos referentes al cliente con su pedido y el total del pedido

### Tabla 24

*Tabla factura*

Campo	PK	FK	Tipo	Descripción
id_fac	•		Integer	Id de la factura
fecha_fac			Date	Fecha de emisión de la factura
cedula_cli		•	Varchar(10)	Número de identificación del cliente
direccion			Varchar(300)	Dirección del cliente
subtotal			Float	Subtotal del pedido
total			Float	Total, del pedido
nombre			Varchar(500)	Nombre del cliente
email			Varchar(200)	Correo electrónico del cliente
cod_pedido		•	Integer	Código del pedido
telefono			Varchar(10)	Teléfono del cliente

*Nota: Tabla factura de la Base de Datos. Elaborado por: Los autores.*

### **2.3.3 Diagramas de Secuencia**

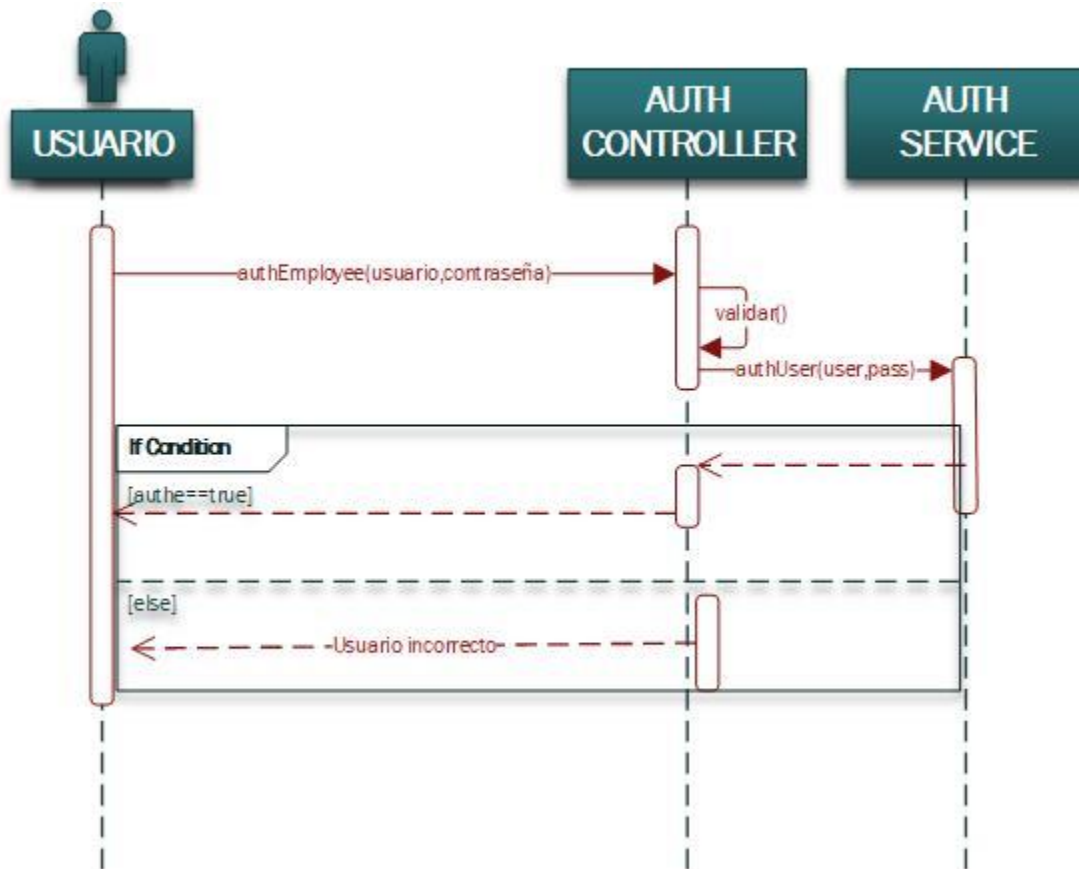
Los diagramas de secuencia representan la forma en cómo un conjunto de actores y objetos en un proceso interactúan a lo largo del tiempo, al igual de mostrar los mensajes entre los participantes y los objetos del proceso y el orden en que se producen. (Cillero, 2020)

A continuación, se muestran los siete diagramas de secuencia realizados por los requerimientos del sistema:

#### **2.1.3.1. Inicio de Sesión**

Para el inicio de sesión, cada actor deberá disponer de un usuario y contraseña se verificará los datos y podrán acceder al sistema. En caso del cliente si es primera vez el sistema le pedirá que ingrese una contraseña para el inicio de sesión.

**Figura 9**  
*Diagrama de secuencia Inicio de sesión*

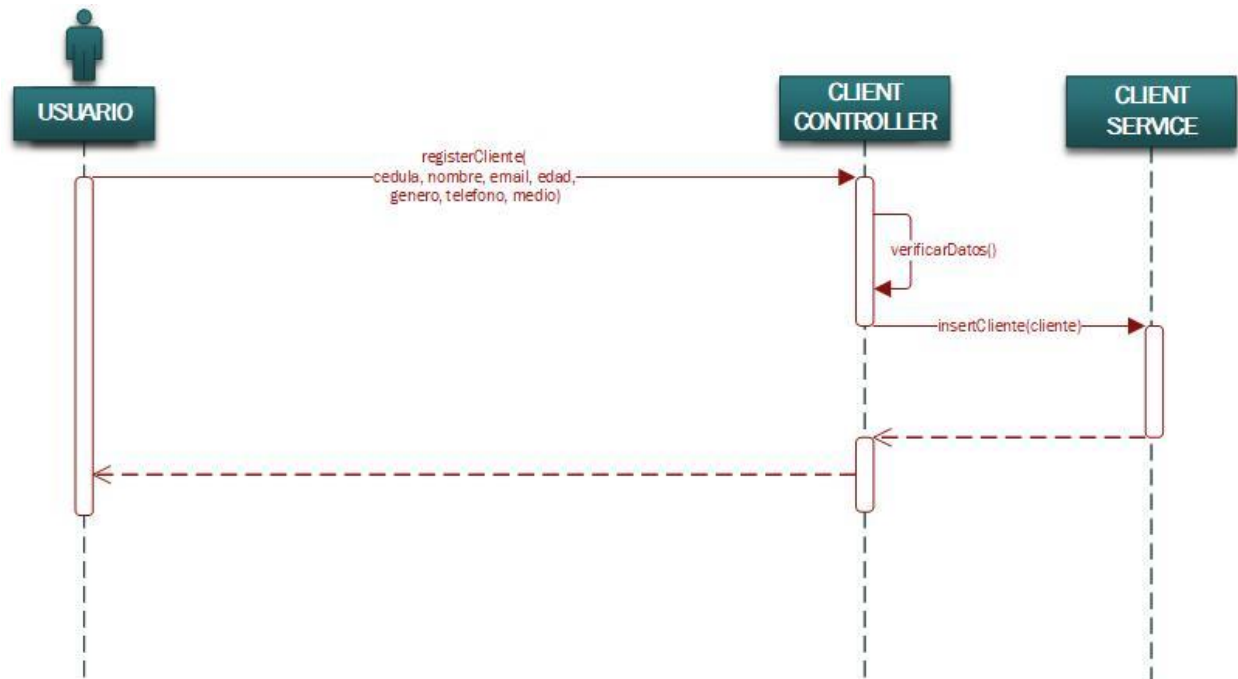


*Nota: Diagrama de secuencia Inicio de sesión. Elaborado por: Los autores.*

### 2.1.3.2. Registrar Cliente

Para el registro del cliente, el administrador o empleado deberá ingresar los datos del formulario que serán proporcionados por el cliente, el sistema verificará que todos los datos sean correctos y devolverá un mensaje de registro exitoso.

**Figura 10**  
*Diagrama de secuencia Registrar Cliente*

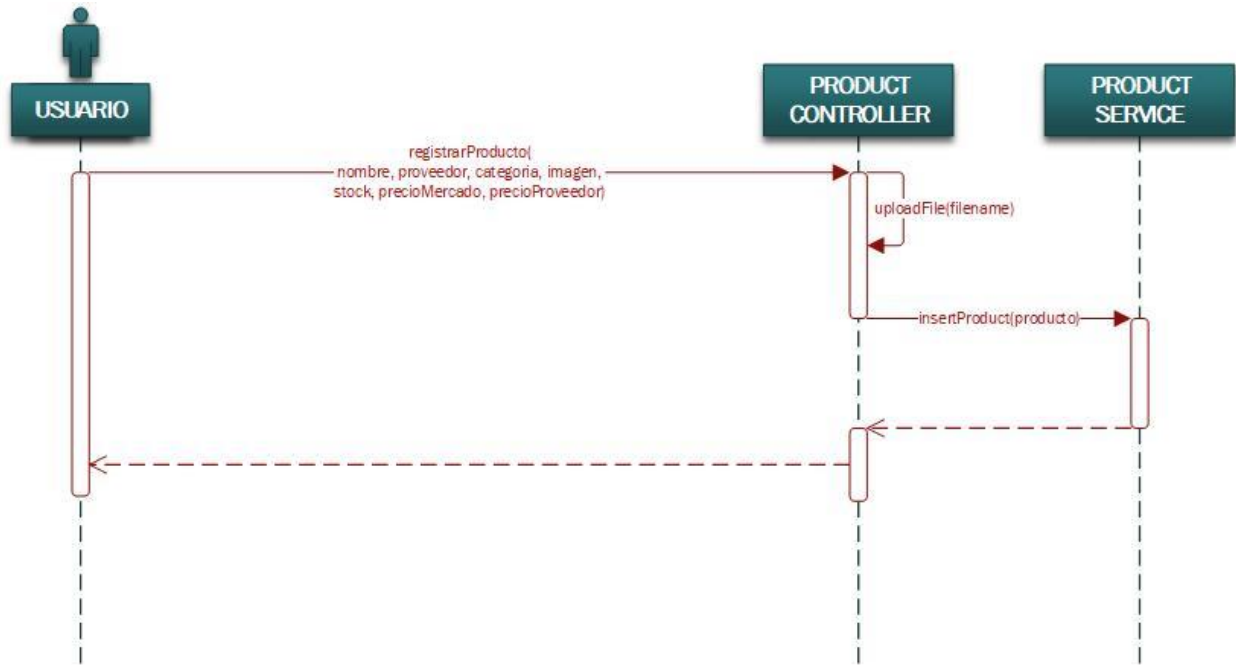


*Nota: Diagrama de secuencia registro del cliente. Elaborado por: Los autores.*

### **2.1.3.3. Registrar Producto**

Para el registro del producto, el administrador o empleado deberá ingresar los siguientes datos del formulario: código del producto, nombre, proveedor, material, peso en kg, tamaño, color, talla, origen, cantidad, precio del mercado, precio del proveedor seguido se guardan los datos en la base de datos y devuelve un mensaje de registro exitoso.

**Figura 11**  
*Diagrama de secuencia Registrar Producto*



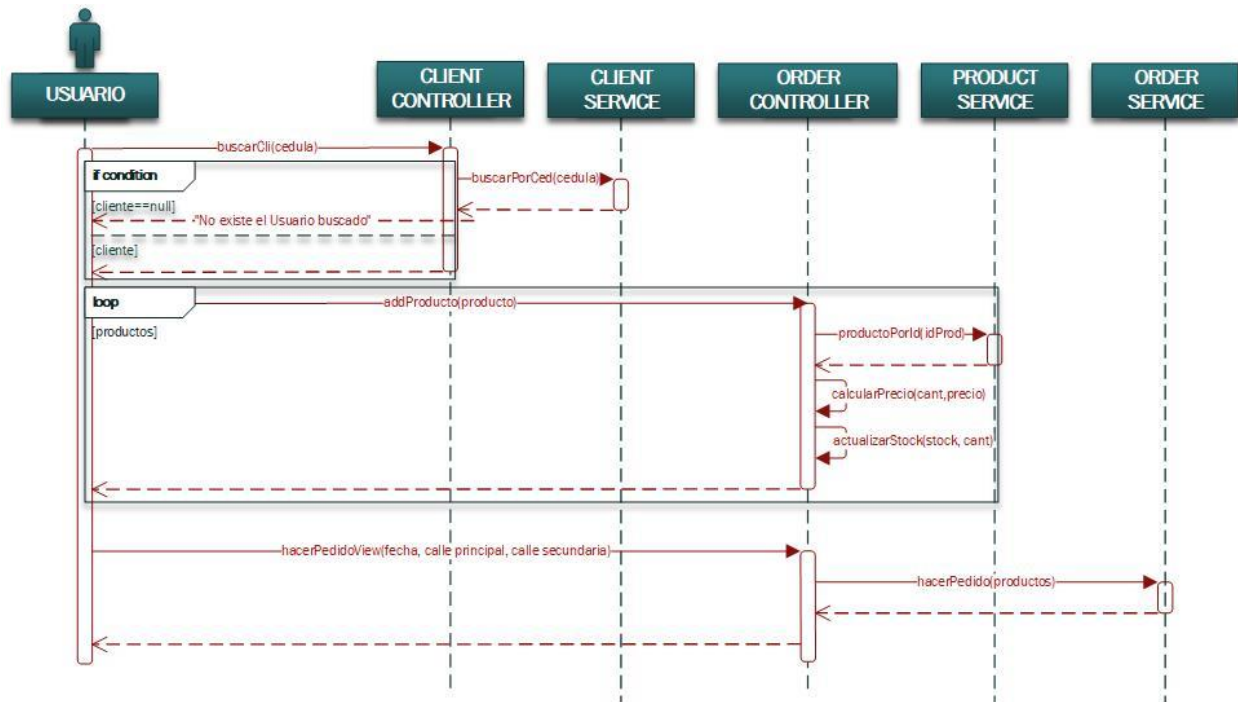
*Nota: Diagrama de secuencia registro del producto. Elaborado por: Los autores.*

#### **2.1.3.4. Registrar Pedido**

Para el registro del pedido, el administrador o empleado deberá ingresar un número de cédula de un cliente que anteriormente haya estado registrado en el sistema, se buscará y se obtendrán los datos del cliente, si no lo encuentra desplegará un mensaje de “No existe el usuario”. Seguido se ingresarán en una lista los productos y la cantidad de cada uno, se calculará el total del pedido y por último se guardará el pedido con un mensaje que devuelve el sistema de “Pedido guardado correctamente”.



**Figura 12**  
*Diagrama de secuencia Registrar Pedido*

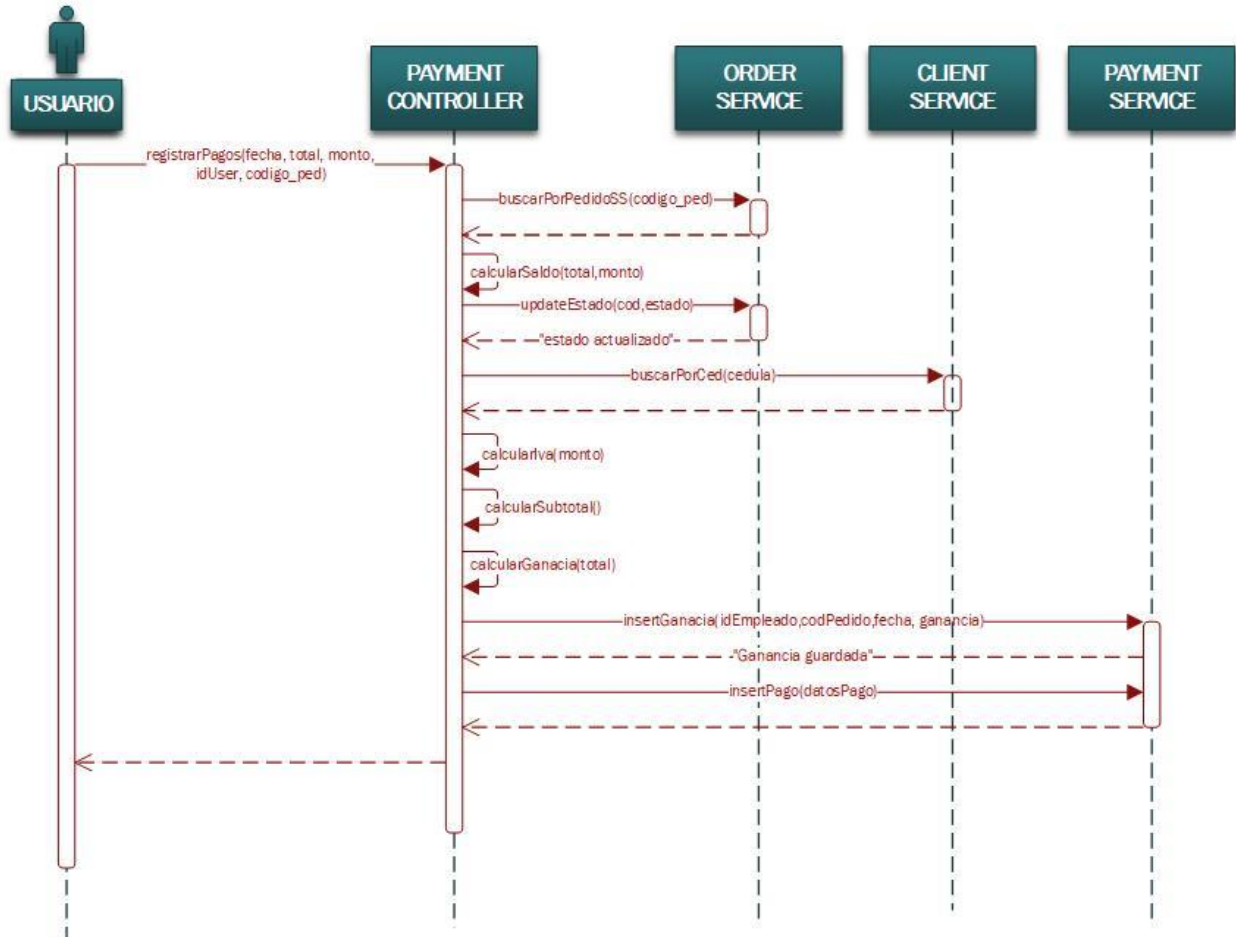


*Nota: Diagrama de secuencia registro del pedido. Elaborado por: Los autores.*

### 2.1.3.5. Registrar Pago

Para el registro de pago, el repartidor al terminar la entrega deberá seleccionar el método de pago, el tipo de pago, el monto recibido por parte del cliente, el cambio y el saldo. El sistema buscará el pedido en la base de datos después calculará el iva y el subtotal del monto recibido para la generación de la nota de venta, al igual que cambiar el estado del pedido al momento de registrar el pago y, por último, se calculará y registrará la ganancia del repartidor por el pedido en el sistema y devolverá un mensaje de “Pago registrado con éxito”.

**Figura 13**  
*Diagrama de secuencia Registrar Pago*

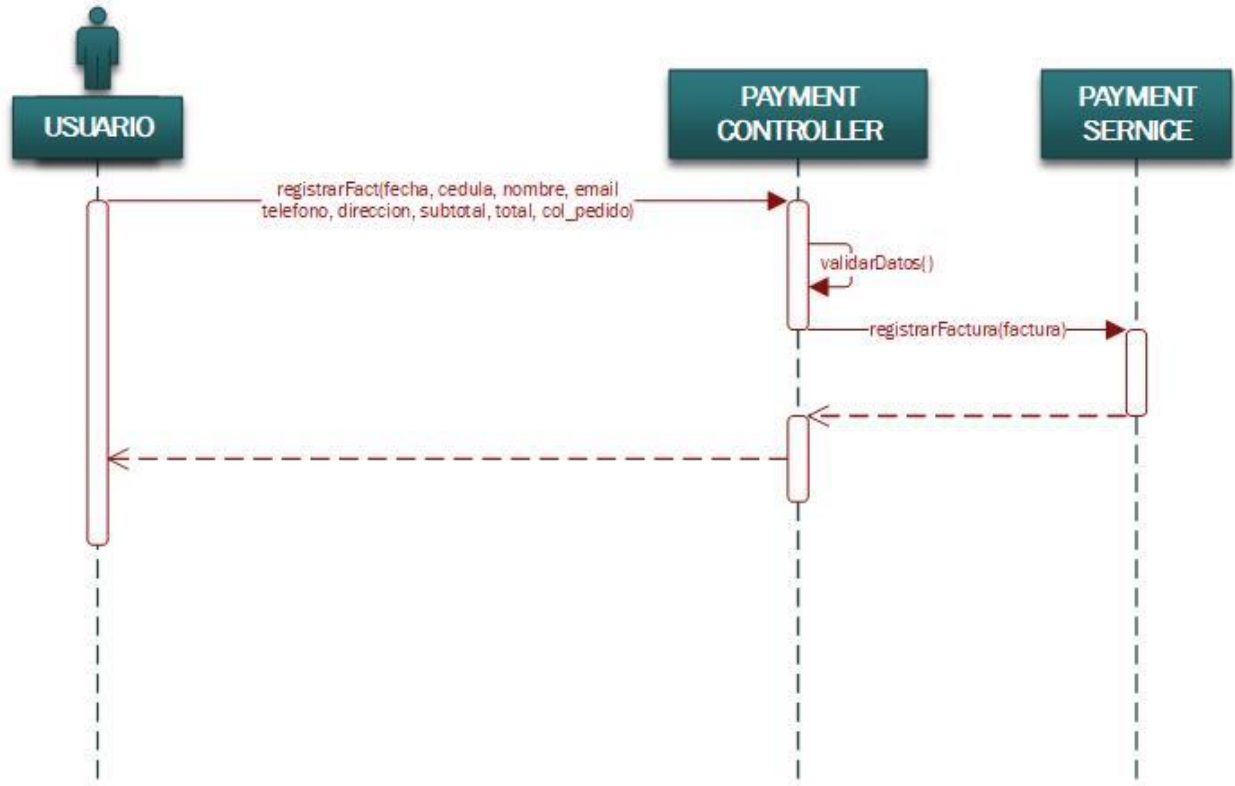


*Nota: Diagrama de secuencia registro del pago. Elaborado por: Los autores.*

**2.1.3.6. Ingreso de datos para nota de venta**

Para la nota de venta después de que el repartidor haya registrado el pago, deberá ingresar los datos de la nota de venta que serán proporcionados por el cliente con el total recibido, se verificará los datos y se procederá a guardarlos con un mensaje del sistema de “Registro de nota de venta exitoso”.

**Figura 14**  
*Diagrama de secuencia Registrar Nota de Venta*

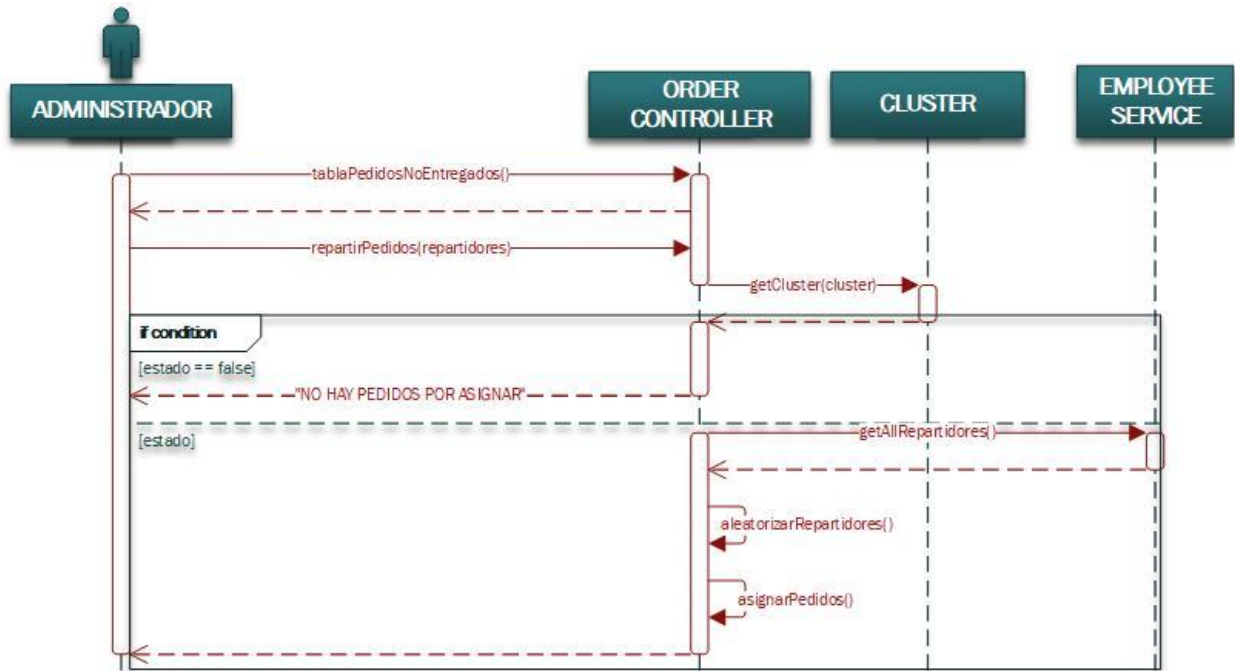


*Nota: Diagrama de secuencia ingreso de datos para la nota de venta. Elaborado por: Los autores.*

### **2.1.3.7. Asignación de pedidos**

Para la asignación de pedidos, el administrador de la tienda deberá ingresar el número de repartidores que se encuentren disponibles en el día, el sistema enviará una petición http a la API la cual obtendrá todos los pedidos registrados del día y los repartirá dependiendo la dirección que haya ingresado el dueño del pedido, después aleatorizará la lista de los repartidores disponibles y los asignará, por último, desplegará un mensaje de "Repartidores asignados correctamente".

**Figura 15**  
*Diagrama de secuencia de Asignación de Pedidos*



*Nota: Diagrama de secuencia de asignación de pedidos. Elaborado por: Los autores.*

### 2.3.4 Diseño de Interfaz

A continuación, se mostrarán el diseño de las principales interfaces de la aplicación creados mediante el software de diseño “Figma” (Bracey, 2018), plasmando un sistema con pantallas que cumplan con los requerimientos establecidos por la tienda.

Para la navegación del sistema se propone un Sidebar colocado al lado izquierdo de cada la interfaz, tal y como se mostrarán en las figuras posteriores. El Sidebar contará con enlaces de navegación variables dependiendo del tipo de usuario que ingrese al sistema, siendo tan solo la página principal o Dashboard el único enlace común para cada usuario.

#### 2.1.3.1. Registro del Cliente

La siguiente interfaz será accesible tanto para el administrador como para los empleados. Cuenta con un formulario sencillo que permite registrar en el sistema los datos más importantes del cliente, mismos datos que posteriormente podrán ser accedidos fácilmente por el administrador o empleados de la tienda.

**Figura 16**  
*Diseño de la Interfaz de Registro del Cliente*

The image shows a web interface for 'in virtual STORE'. On the left is a dark blue sidebar with 'Dashboard' and 'Menú Principal' links. The top right has a light blue header with 'User Name'. The main content area is titled 'REGISTRO DEL CLIENTE' and contains a registration form with the following fields: 'Nombres:', 'Cédula:', 'E-mail:', 'Edad:', 'N° de Teléfono:', and 'Medio de compra:' (a dropdown menu). A dark blue 'REGISTRAR' button is positioned below the form.

*Nota: Diseño de la Interfaz de Registro del Cliente. Elaborado por: Los autores.*

### **2.1.3.2. Registro de Productos**

La siguiente interfaz será accesible tanto para el administrador como para los empleados. Contará con un formulario con varios campos que permitirá ingresar los datos necesarios de los productos que deseen registrar.

Adicionalmente el formulario permite la carga de una fotografía del producto, misma que puede ser previsualizada dentro de la interfaz al lado derecho de la pantalla, con el fin de asegurarse de cargar la imagen correcta antes de ser guardada por el usuario.

### Figura 17

*Diseño de Interfaz de Registro de un Producto*

The interface is titled "REGISTRA UN PRODUCTO" and is part of the "in virtual STORE" system. It features a sidebar with "Dashboard" and "Menú Principal" options. The main form includes fields for product name, provider, weight, size, color, optional size, origin, and stock quantity. It also has fields for market and provider prices, a "Subir Imagen" button, and a "Imagen del Producto" placeholder. A "GUARDAR" button is located at the bottom of the form.

*Nota: Diseño de la Interfaz de Registro de un Producto. Elaborado por: Los autores.*

#### 2.1.3.3. Registro de Pedido

Esta interfaz es parte central del proceso de la empresa, ya que permite registrar los pedidos realizados por el cliente. En primer lugar, cuenta con un formulario de búsqueda de los datos del cliente mediante su cédula o un enlace que lo lleve a registrar un nuevo usuario.

Una vez generada la búsqueda, se desplegará los datos del cliente al lado izquierdo de la pantalla, con el objetivo de que la información del cliente pueda ser comprobada previo a la generación del pedido. Mientras que al lado derecho de la pantalla se muestra un formulario donde se lista los productos disponibles.

El formulario permite escoger un producto y colocar la cantidad a comprar para ser agregado al pedido, cada producto agregado se va mostrando en formato de resumen en una tabla ubicada bajo el formulario. Finalmente, la persona que esté registrando el pedido revisa la tabla de resumen para comprobar que el pedido sea correcto y lo registra en el sistema.

**Figura 18**  
*Diseño de Interfaz de Registro de un Pedido*

The interface is titled "REGISTRO DEL PEDIDO". It includes a sidebar with "Dashboard" and "Menú Principal". The main area has a search form with "Cédula:" and "Producto:" fields, and buttons for "BUSCAR", "NUEVO CLIENTE", and "AÑADIR". A table with columns "Producto", "Cantidad", and "Precio Total" is present, along with a "Datos cliente" box and a "Hacer Pedido" button.

*Nota: Diseño de Interfaz de Registro de un Pedido. Elaborado por: Los autores.*

#### 2.1.3.4. *Asignación de Pedidos*

La siguiente interfaz es de uso exclusivo del administrador del sistema. En la parte superior de la pantalla se muestra una tabla con los pedidos que su fecha de entrega está establecida para el día siguiente.

Con los datos mostrados en la tabla, el administrador cuenta con un pequeño formulario sobre la tabla, donde ingresa el número de repartidores disponibles para dicho día y da clic en repartir los pedidos según el número de repartidores.

Mientras tanto en la parte inferior de la pantalla se muestra una tabla con los pedidos que cuentan con fechas de entrega posteriores ordenadas por fecha, con el fin de que el administrador tenga conocimiento de los siguientes pedidos de la tienda.

**Figura 19**  
*Diseño de Interfaz de Asignación de Pedidos*

The interface is titled "ASIGNAR PEDIDOS". It includes a sidebar on the left with "Dashboard" and "Menú Principal" options. The top right corner shows the "User Name". The main content area contains a form for "Número de Repartidores:" with a text input field and a "REPARTIR" button. Below this are two tables. The first table has columns for "PEDIDO", "FECHA DE ENTREGA", and "ESTADO". The second table, titled "FUTURAS ENTREGAS", also has columns for "PEDIDO", "FECHA DE ENTREGA", and "ESTADO".



*Nota: Diseño de Interfaz de Asignación de Pedidos. Elaborado por: Los autores.*

### **2.1.3.5. Registro de Pagos**

La siguiente interfaz, es accesible tanto para el administrador, empleado y repartidor. Esta interfaz es parte vital del sistema ya que permitirá registrar los pagos realizados por los clientes.

Esta pantalla será accesible tanto para la primera vez que el usuario realiza el pago del pedido, así como también podrá a través de un botón de actualización para los casos en los que el cliente haya optado por el pago por cuotas.

La pantalla en la parte superior mostrará el código del pedido del cual se está realizando el registro, a continuación, cuenta con un formulario que permitirá ingresar las opciones de pago escogidas por el cliente. Y finalmente, en la parte inferior mostrará el saldo pendiente del pedido.

#### **Figura 20**

*Diseño de Interfaz de Registro de Pagos*

The image shows a web interface for recording payments. It features a blue sidebar on the left with the 'in virtual' logo and navigation links for 'Dashboard' and 'Menú Principal'. The main content area has a light blue header with a 'User Name' field. The central heading is 'REGISTRAR PAGOS'. Below this, there is a label 'Pedido: (Código Pedido)'. The form consists of four input fields: 'Total a Pagar:', 'Método de pago:' (which includes a dropdown arrow), 'Monto Recibido:', and 'Saldo:'. At the bottom of the form is a dark blue button labeled 'REGISTRAR'.

*Nota: Diseño de Interfaz de Registro de Pagos. Elaborado por: Los autores.*

### 2.1.3.6. *Detalle del Pedido*

La siguiente interfaz se accesible tanto para el administrador, empleados, y clientes; la interfaz presentará ligeras variaciones en la visualización de un cliente.

La pantalla mostrará el código del pedido al cual se accedió. A continuación, se muestra una tabla con el resumen del pedido realizado listando cada producto. Finalmente, en la parte inferior se muestra el costo del pedido, así como el saldo pendiente que pueda tener cada cliente.

En la parte superior solo los empleados de la tienda y el administrador del sistema cuentan con dos campos extras donde podrán visualizar el nombre del cliente y el teléfono de este, con el fin de puedan contactar al cliente en casos de deudas pendientes.

**Figura 21**  
*Diseño de Interfaz de Visualización del Detalle del Pedido*

The interface is divided into a sidebar and a main content area. The sidebar is dark blue and contains the logo 'in virtual STORE', 'Dashboard', and 'Menú Principal'. The main content area has a light blue header with 'User Name' and a title 'DETALLE DEL PEDIDO'. Below the title are input fields for 'Cliente:', 'Teléfono:', 'Pedido: (Código del Pedido)', 'Costo del Pedido:', and 'Saldo Pendiente:'. A table with three columns: 'Producto', 'Cantidad', and 'Precio Total' is displayed below the input fields.

Producto	Cantidad	Precio Total

*Nota: Diseño de Interfaz de Visualización del Detalle del Pedido. Elaborado por: Los autores.*

### 2.1.3.7. Creación Nota de Venta

La siguiente interfaz será accesible para la persona que registre el pago completo, es decir puede ser el administrador, cualquier empleado de la tienda o el repartidor al que fue asignado a un pedido.

Esta pantalla está destinada para crear una nota de venta. misma que se podrá generar únicamente cuando el cliente haya realizado el pago completo del pedido.

Cuenta con un formulario que tiene los datos de los clientes precargados, y el detalle del pedido en una tabla con su precio final. Una vez verificado los datos, la nota de venta cuenta con el botón de guardado, registrándola en el sistema, una vez guardada la nota de venta se vuelve accesible de forma permanente tanto para el administrador como para cliente.

**Figura 22**  
*Diseño de Interfaz de Creación de Nota de Venta*

The interface is divided into a sidebar and a main content area. The sidebar is dark blue and contains the logo 'in virtual STORE' and two menu items: 'Dashboard' and 'Menú Principal'. The main content area is light blue and features a form titled 'NOTA DE VENTA'. The form includes five input fields: 'Fecha:', 'Nombres:', 'Cédula/RUC:', 'Dirección:', and 'Teléfono:'. Below the form is a table titled 'DETALLES' with three columns: 'Nombre del Producto', 'Cantidad', and 'Precio Total'. At the bottom of the form, there is a 'Precio Final:' label followed by an input field and a dark blue button labeled 'GUARDAR'.

*Nota: Diseño de Interfaz de Visualización del Detalle del Pedido. Elaborado por: Los autores.*

### 2.1.3.8. Seguimiento del Pedido

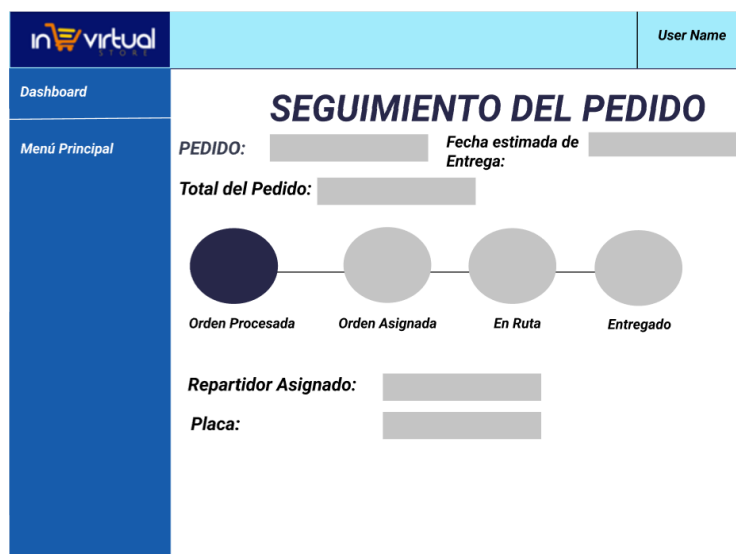
La siguiente interfaz es de uso exclusivo para el cliente. Esta pantalla muestra al cliente el estado del pedido realizado, con el objetivo de que tenga información actualizada del mismo.

La pantalla, en la parte superior muestra la información más relevante del pedido, mientras que en la parte central se despliega un gráfico dinámico que se irá marcando sus campos dependiendo del estado del pedido.

Finalmente, en la parte inferior de la pantalla, una vez el pedido sea asignado a un repartidor se mostrará el nombre del repartidor al que ha sido designado la entrega de dicho producto, con la respectiva placa del transporte en el que se dirige. Este campo es útil para que en caso de entregas dentro de la ciudad de Quito los clientes cuenten con más seguridad al momento de retirar el pedido y realizar su pago.

#### Figura 23

*Diseño de Interfaz de Seguimiento de un Pedido*



*Nota: Diseño de Interfaz de Visualización del Detalle del Pedido. Elaborado por: Los autores.*

### **3. CONSTRUCCIÓN Y PRUEBAS**

#### **3.1 CONSTRUCCIÓN**

El siguiente subcapítulo mostrará el proceso de construcción de la aplicación web de InVirtual Store. Se explicará y se justificará el uso de las tecnologías de Node.js y HandleBars.js para su desarrollo, así como el uso en conjunto de las metodologías de gestión Agile, Scrum y Kanban.

##### ***3.1.1 Metodología de Desarrollo***

Para el desarrollo de la aplicación se utilizaron dos de las metodologías Ágiles más utilizadas en la gestión de proyectos, SCRUM como metodología principal, y Kanban de manera complementaria.

Se ha escogido la metodología Scrum, con la finalidad de lograr un flujo de trabajo bien definido en cada uno de los objetivos establecidos para cada Sprint, de una manera ágil y efectiva con cada entregable.

De manera complementaria, se ha utilizado a la metodología de Kanban, ya que permite contar con una mejor visión en los diversos flujos de trabajo. Este método está basado en tarjetas que contienen cada tarea necesaria para el proyecto. Kanban funciona como una fuente de información, ya que muestra posibles cuellos de botella durante el proceso e impedimentos hacia un flujo de trabajo continuo e ininterrumpido.

##### ***3.1.2 Product Backlog***

Dentro de la metodología Scrum, el Product Backlog funciona como eje central del desarrollo. Este concepto se basa en proveer un listado de las tareas que están planificadas, y van a realizarse a lo largo del ciclo de vida del proceso, con el fin de cumplir con los requerimientos del sistema de forma prioritaria. De igual manera el Product Backlog puede cambiar con el transcurso del

proyecto, con el surgimiento de nuevas funcionalidades no previstas, o errores (bugs), que afecten al sistema.

El Product Backlog, se muestra mediante una tabla con la información requerida para describir las actividades a realizar. La tabla cuenta con 4 columnas explicadas a continuación:

**Tabla 25**  
*Descripción Product Backlog*

<b>ID:</b>	Identificador de la tarea que se va a realizar.
<b>Tarea Asignada:</b>	Descripción sintetizada de la actividad a realizar.
<b>Puntos de Dificultad:</b>	Estimación relativa de la dificultad, utilizando la escala de la secuencia exponencial de Fibonacci. Para este proyecto se usará solo una pequeña escala de 3 posibles puntos, siendo 1 el mínimo nivel de dificultad, y 3 el máximo. (Sandeep et al., 2022)
<b>Nivel de Prioridad:</b>	Estimación del nivel de prioridad que tiene la actividad, según acuerdos llegados tanto entre el equipo de desarrollo como con todas las partes interesadas (stakeholders) en 3 niveles de dificultad: Alto, Medio y Bajo.

*Nota: Descripción de las características del Producto Backlog. Elaborado por: Los autores.*

**Tabla 26**  
*Tabla de Product Backlog*

<b>ID</b>	<b>Tarea Asignada</b>	<b>Puntos de Dificultad</b>	<b>Nivel de Prioridad</b>
1	Creación de la BBDD y carga de Datos iniciales	1	Alto
2	Creación del entorno de ejecución de la aplicación y conexión a la Base de Datos	2	Alto
3	Subir Proyecto a Repositorio en GitHub	1	Alto
4	Creación de sesiones del sistema, separados por cada tipo de usuario.	3	Medio
5	Implementación de la Navegación del Sistema	2	Medio
6	Creación de elementos del Layout según cada tipo de usuario.	1	Medio
7	Implementación de Registro de Empleados.	1	Medio
8	Implementación de registro de Productos	1	Alto
9	Implementación de registro de Clientes	1	Medio
10	Implementación del registro de Pedidos	3	Alto
11	Implementación del registro de pagos de clientes.	3	Alto
12	Implementación de generación de Nota de Venta	1	Bajo

13	Cambio automático del Estado de Pedido del Cliente	3	Medio
14	Creación de Tablas de resumen de cada ítem para el Administrador.	1	Medio
15	Permitir la edición y eliminación de campos desde la ventana del administrador	2	Medio
16	Creación de tablas de Detalle de Pedidos y Deudas para Administrador y Cliente	2	Alto
17	Creación de Script en Python, para realizar Clustering según la ubicación de los pedidos.	3	Medio
18	Creación de API Rest para vincular Script de Python con el sistema principal	3	Medio
19	Gestionar la lista de Pedidos asignados a cada repartidor.	2	Bajo
20	Validaciones de datos	1	Medio
21	Configuración del Entorno de Integración Continua (CI)	3	Bajo

*Nota: Tabla Sprints de las tareas asignadas, Puntos de dificultad y Nivel de Prioridad.  
Elaborado por: Los autores.*

### **3.1.3 Sprints**

Dentro de esta metodología, el Sprint es la unidad básica de trabajo del Scrum Team. Esta es la característica clave que distingue a Scrum de otros modelos de desarrollo ágil. El sprint es una iteración simple realizada por los miembros del equipo, se puede realizar múltiples Sprints durante el desarrollo del proyecto.

El Sprint comienza con un compromiso de equipo y finaliza con una demostración de los resultados. El tiempo mínimo de sprint es de una semana, el máximo es de 4 semanas (Lara Walter, 2015). Para este proyecto se ha decidido que cada Sprint tendrá una duración de 2 semanas.

Para poder llevar un registro de las actividades realizadas en cada Sprint es necesarios contar un con Sprint Backlog definido en cada uno con el objetivo de mantener los objetivos y las tareas correctamente definidas en cada Sprint. (Sudarsono, 2020)

Para visualizar de mejor manera el Sprint Backlog, se ha recurrido a la utilización de un tablero Kanban que registra las tareas completadas, el responsable y el estado para cada tarea. El tablero Kanban cuenta con 3 columnas:

- **To Do:** Son las tareas designadas en el Sprint que están por realizarse
- **In Progress:** Las tareas que están siendo trabajadas por el desarrollador designado
- **Done:** La tarea que ha sido completada y ha cumplido con los Criterios de Aceptación (AC)

Cada nota colocada dentro del tablero cuenta con un título general, una descripción corta de la tarea a realizar, y los criterios de aceptación (A.C.). Los A.C. son una lista de requerimientos específicos que son necesarios cumplirlos para considerar a la tarea como terminada. (Lucassen et al., 2016)

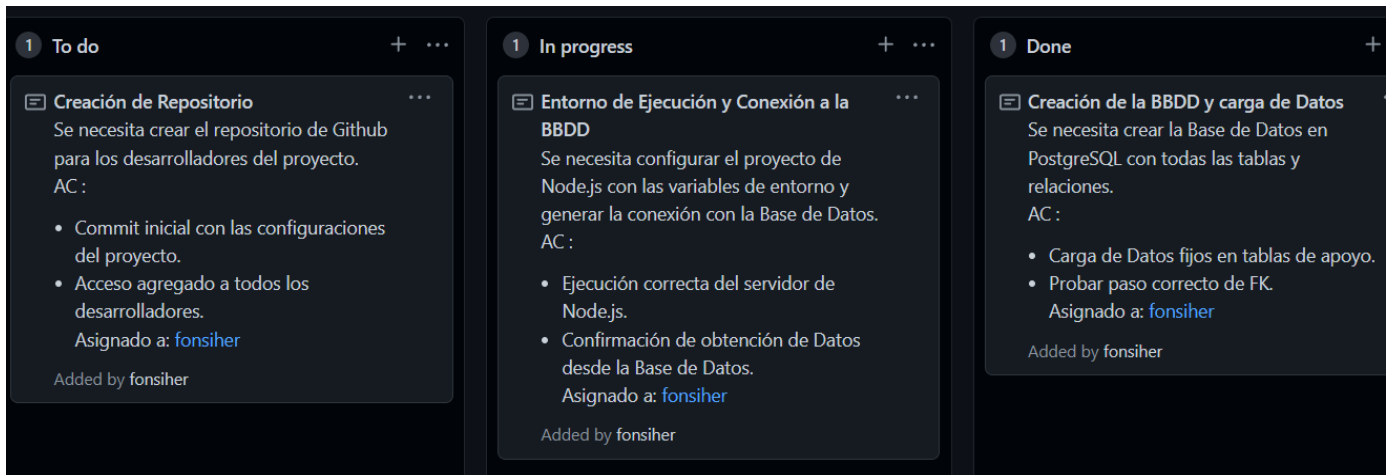
#### **4.1.3.1. *Sprint 1: Configuración inicial y creación de entorno de ejecución.***

**Objetivo del Sprint:** Realizar la configuración inicial del proyecto, mediante la activación y configuración de las tecnologías tanto de Back-End como de Base de Datos, para lograr establecer los parámetros iniciales a seguir en el desarrollo de la aplicación.



## Figura 24

### Tablero Kanban durante el Sprint 1



*Nota: Tablero Kanban durante el Sprint 1. Elaborado por: Los autores.*

- **Creación de la BBDD y Carga de Datos.**

- Se crea la Base basado en el motor PostgreSQL, se utilizó la aplicación pgAdmin4, para la administración de las bases de Datos con una interfaz gráfica. Adicionalmente se cargó dentro de la base de datos creadas una query en SQL, que es el generado mediante el propio programa de modelamiento utilizado, Power Architect según el modelo relacional realizado:

**Figura 25**  
Carga del Script Inicial

```

InVirtual/postgres@PostgreSQL 9.6
Query Editor Query History
1
2 CREATE SEQUENCE public.categoria_producto_id_cat_seq;
3
4 CREATE TABLE public.CATEGORIA_PRODUCTO (
5     ID_CAT INTEGER NOT NULL DEFAULT nextval('public.categoria_producto_id_cat_seq'),
6     CATEGORIA VARCHAR(200) NOT NULL,
7     CONSTRAINT categoria_producto_pk PRIMARY KEY (ID_CAT)
8 );
9
10
11 ALTER SEQUENCE public.categoria_producto_id_cat_seq OWNED BY public.CATEGORIA_PRODUCTO.ID_CAT;
12
13 CREATE SEQUENCE public.genero_id_gen_seq;
14
15 CREATE TABLE public.GENERO (
16     ID_GEN REAL NOT NULL DEFAULT nextval('public.genero_id_gen_seq'),

```

*Nota: Carga del SQL Script inicial de la Base de Datos. Elaborado por: Los autores.*

- Se cargan datos que serán estáticos y serán necesarios para la generación de relaciones de las demás tablas, se cargan los datos de las tablas: genero, tipo\_employado y categoría\_producto:

**Figura 26**  
Carga de tablas con datos estáticos

Data Output		Explain	Messages	Notifications	Data Output		Explain	Messages	Notifications
<b>id_cat</b> [PK] integer	<b>categoria</b> character varying (200)				<b>id_tipo</b> [PK] integer	<b>nombre_tipo</b> character varying			
1	1 Ropa				1	1 Administrador			
2	2 Cosméticos				2	2 Empleado			
3	3 Perfumes				3	3 Repartidor			
4	4 Salud								


*Nota: Carga de datos estáticos en las diferentes tablas de la base de datos. Elaborado por: Los autores.*

- **Entorno de Ejecución y Conexión a la BBDD**

- Para el siguiente trabajo se creó el proyecto utilizando el entorno de Node.js v16.17.0 y como gestor de paquetes a Node Package Manager (npm) v8.18.0.
- Se creó el directorio donde se encuentra el proyecto, en conjunto con un archivo central llamado *app.js*. Para iniciar el proyecto, y pueda utilizar dependencias externas se ejecuta el comando *\$npm init*.
- Se instalaron las dependencias y librerías principales que tiene la aplicación. La primera es express, que es un framework web minimalista, rápido y sin supervisión para Node, que permite la conexión a un servidor. Y **hbs**: Esta librería funciona como motor de visualización predeterminado requiere solo una línea de código en la configuración de su aplicación, para handlebars.js.
- Dentro del *package.json* se colocó la información inicial del proyecto, la configuración de los scripts de ejecución, y las dependencias instaladas. Una vez guardada la configuración el proyecto se lo puso inicializar mediante el comando *\$npm start*.
- Se creó el archivo de conexión a la base de Datos, mediante la utilización de la librería **pg**. Esta librería es un cliente PostgreSQL sin bloqueo para Node.js, JavaScript puro y enlaces libpq nativos opcionales. Se configura colocando las credenciales que apuntan a la Base de Datos:

## Figura 27

### Conexión local a la Base de Datos

```
src > model > connection >  conexion.js > ...
1  const { Pool } = require("pg");
2
3  const config = {
4    // Conexión para la base de datos local
5    user: "postgres",
6    host: "localhost",
7    password: "sololdu",
8    database: "InVirtual",
9  };
10 const pool = new Pool(config);
11
12 module.exports = { pool };|
```

*Nota: Conexión local a la Base de datos PostgreSQL. Elaborado por: Los autores.*

- Se configuró el archivo principal especificando la conexión, las rutas del sistema y el gestor a desarrollarse.

## Figura 28

### Inicio de la Aplicación

```
32
33 // Rutas de la página web
34 const routes = require("../src/controllers/routes/routes");
35 app.use(routes);
36
37 app.listen(port, () => {
38   console.log("Servidor Iniciado, escuchando el puerto 3000");
39 });|
```

*Nota: La aplicación inicia un servidor escuchando el puerto 3000. Elaborado por: Los autores.*

- **Creación de Repositorio**

- Se crea un repositorio privado, utilizando la interfaz proveída por Github.

- Se siguieron los comandos proveídos por GitHub, mismos que fueron agregados en el terminal del proyecto para subir el primer commit, y vincular el proyecto al repositorio remoto.

**Figura 29**  
*Repositorio Github*

```
git add .  
git commit -m "first commit"  
git branch -M main  
git remote add origin  
https://github.com/fonsiher/InVirtual-Store.git  
git push -u origin main
```

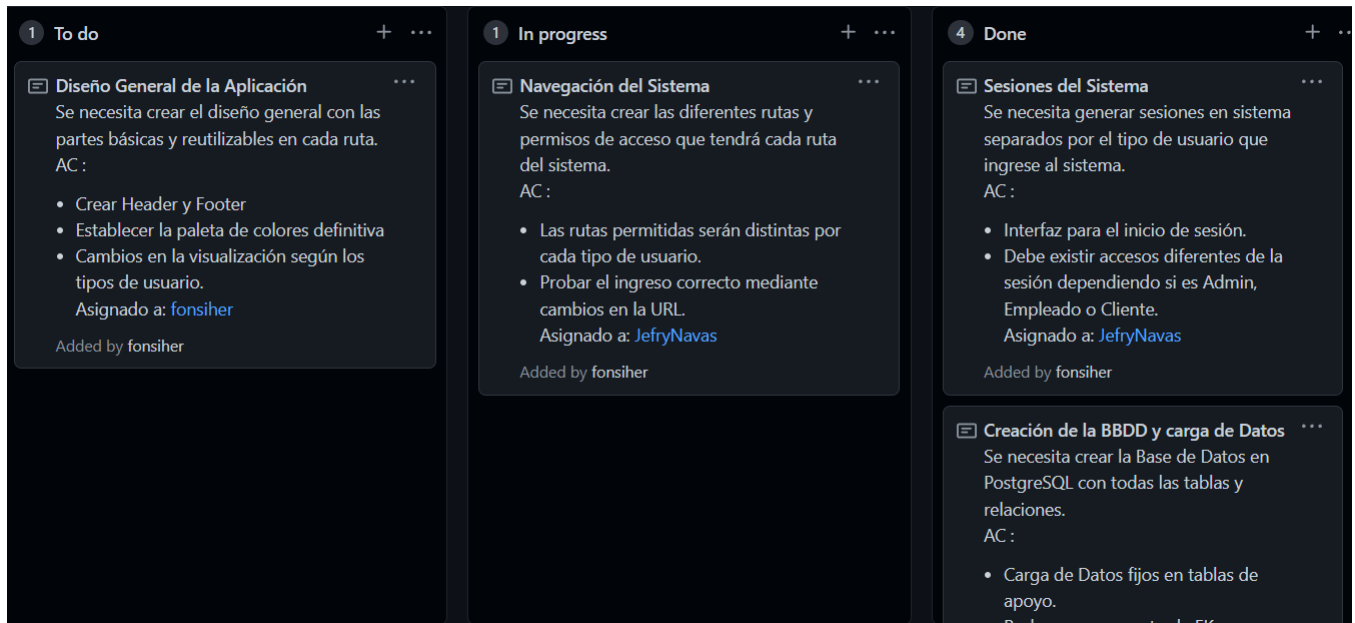
*Nota: Código de inicialización del repositorio en Github. Elaborado por: Los autores.*

#### **4.1.3.2.      *Sprint 2: Sesiones y Navegación de la Aplicación***

**Objetivo del Sprint:** Implementar las sesiones de usuario del sistema dependiendo de los cuatro tipos de usuario con sus respectivos permisos y restricciones en su acceso, mediante utilización de librerías especializadas, y creación de reglas tanto a nivel de Front-End como de Back-end, para tener organizado el esqueleto de la aplicación según cada tipo de usuario.

**Figura 30**

*Tablero Kanban durante el Sprint 2*



*Nota: Tablero Kanban durante el Sprint 2. Elaborado por: Los autores.*

- **Sesiones del Sistema**

- Para conseguir gestionar las sesiones del sistema se recurrió al uso de la librería: **express-session**. Esta librería crea un middleware de sesión con el archivo options, y permite la navegación de sesiones en el servidor evitando posibles errores y amenazas que pueden ocasionar el uso del Local Storage, u otras librerías como JWT (JSON Web Token).

**Figura 31**

*Sesiones en la Aplicación*

```
// express session
app.use(
  session({
    secret: "secret",
    resave: true,
    saveUninitialized: true,
  })
);
```

*Nota: Configuración de sesiones en app.js. Elaborado por: Los autores.*

- Una vez configurada e instanciada la librería en el archivo principal. Se crearon las consultas con la ayuda de la librería pg, que busca la existencia del usuario de acuerdo con las credenciales. Si la consulta es exitosa devuelve los datos del usuario, caso contrario envía un mensaje de error.

### Figura 32

#### Obtención del usuario logeado

```
const { pool } = require("../model/connection/conexion")
const authUser = async(email, pass) => {
  try {
    const res = await pool.query(`select * from empleado where email='${email}' and password = '${pass}'`);
    if (res.rows == 0) {
      return false;
    }
    return res.rows;
  } catch (error) {
    return error.message;
  }
}
```

*Nota: Query para extraer los datos del usuario logeado. Elaborado por: Los autores.*

- Adicionalmente a la validación de existencia del usuario, en un controlador, se procede a verificar con los datos el tipo de usuario, y a través de condicionales, se asigna un variable de sesión distinta a cada uno. Además, se establecen los parámetros que serán leídos a lo largo de la sesión.

### Figura 33

Asignación de sesiones por perfil de usuario

```
let Ruta;
if (authe[0].id_tipo == "2") {
  req.session.loggedinEmpleado = true;
  Ruta = "empleado";
} else if (authe[0].id_tipo == "3") {
  req.session.loggedinRepartidor = true;
  Ruta = "repartidor";
} else if (authe[0].id_tipo == "1") {
  req.session.loggedinAdmin = true;
  Ruta = "admin";
}
req.session.user = {
  nombre: authe[0].nombre,
  email: authe[0].email,
  tipo: authe[0].id_tipo,
  id: authe[0].id_empleado,
};
```

*Nota: Condicionales para asignación de sesión por perfil del usuario. Elaborado por: Los autores.*

- **Navegación Del Sistema**

- Dentro de la librería de la librería express se extrajo la función Router, que permitió generar las rutas del sistema, asociadas a funciones específicas que ejecutan las vistas del sistema.

### Figura 34

Rutas de la Aplicación

```
router.get("/", login);
router.get("/admin", admin);
router.get("/empleado", empleado);
router.get("/repartidor", repartidor);
router.get("/cliente", cliente);
router.get("/producto", producto);
router.get("/pedido", pedido);
router.get("/proveedor", proveedor);
router.get("/register", register);
```

*Nota: Rutas creadas para la aplicación. Elaborado por: Los autores.*



- Las funciones de los controladores que son llamados por las rutas son comparadas a las variables de sesión mediante un condicional, con el fin de restringir controlar el acceso a ciertas rutas de acuerdo con el tipo de usuario que ingresó al sistema.

### Figura 35

Controlador de Ruta de un perfil

```
const admin = function (req, res) {  
  if (req.session.loggedinAdmin) {  
    let user = req.session.user;  
    res.render("admin", {  
      login: true,  
      titulo: "Dashboard",  
      tipo: "admin",  
      name: user.nombre,  
    });  
  }  
};
```

*Nota: Controlador de Ruta con acceso Exclusivo para el administrador del sistema. Elaborado por: Los autores.*

### Figura 36

Controlador de Ruta de varios perfiles

```
const pagos = function (req, res) {  
  if (req.session.loggedinRepartidor || req.session.loggedinAdmin) {  
    let user = req.session.user;  
    res.render("pagos", {  
      login: true,  
      titulo: "Pagos",  
      tipo: user.tipo,  
      name: user.nombre,  
    });  
  } else {  
    res.redirect("/");  
  }  
};
```

*Nota: Controlador de Ruta con acceso a 2 perfiles de usuarios. Elaborado por: Los autores.*

- **Diseño General de la Aplicación.**

- En esta tarea se realizaron los elementos visuales principales de la aplicación, mismos que tienen un esqueleto común, como son el Encabezado, el panel principal, la barra de navegación y el pie de página.

- El Encabezado es simple, ya que tan solo muestra el nombre del usuario, y un menú desplegable, mientras que el pie de página contiene el nombre de la empresa, en conjunto con el año actual.
- El panel principal presenta cambios según el tipo de usuario que ingrese, tanto con el texto de Bienvenida y su imagen central.
- Para la barra de navegación, se realizó un SideBar colocado al lado izquierdo de la pantalla principal. Para crear este menú, se utilizó HTML, CSS, iconos y algunos elementos de Bootstrap para conseguir el diseño responsivo. Esta función se ha integrado para permitir a los usuarios mediante su perfil del sistema moverse de forma rápida y precisa entre las pantallas.

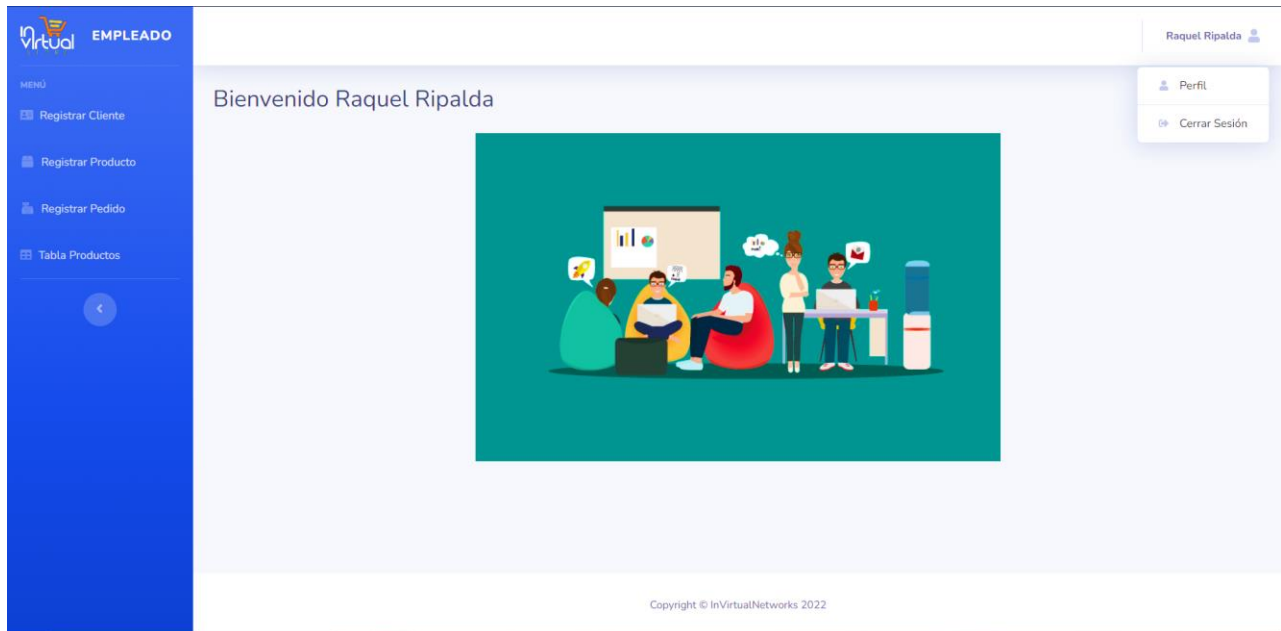
**Figura 37**  
*Pantalla Inicial del Administrador*



*Nota: Pantalla inicial para el Administrador del Sistema. Elaborado por: Los autores.*

## Figura 38

### *Pantalla inicial de un empleado*



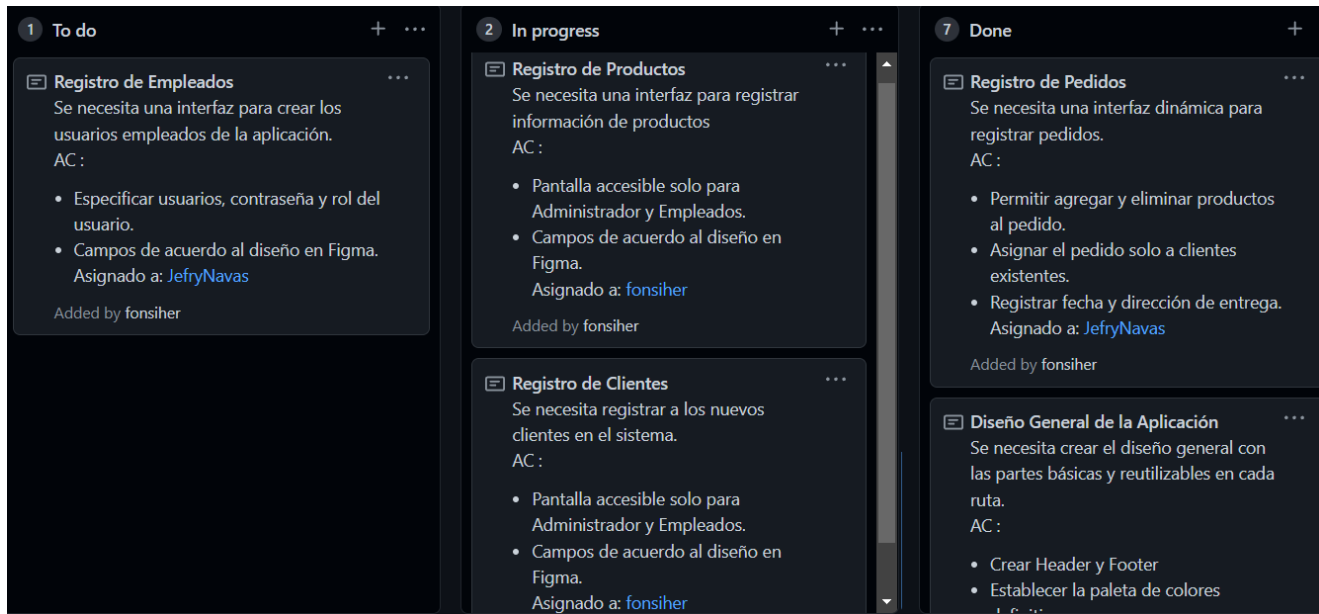
*Nota: Pantalla inicial de un empleado de la tienda. Elaborado por: Los autores.*

#### **4.1.3.3. *Sprint 3: Interfaces de Registro de Datos para el Administrador y Empleados***

**Objetivo del Sprint:** Implementar las diversas interfaces, cada una con formularios específicos según los parámetros abstraídos en la etapa de diseño que permitan guardar los datos necesarios tanto de usuarios, clientes, productos y pedidos, a través de peticiones HTTP (GET y PUT), y configuración correcta de cada formulario, para lograr datos que son guardados en el sistema sean persistentes, al estar respaldadas en la Base de Datos, y se complementen unos de otros.

## Figura 39

### Tablero Kanban durante el Sprint 3



*Nota: Tablero Kanban durante el Sprint 3. Elaborado por: Los autores.*

- **Registro de Empleados**

- El Registro de los empleados se realiza dentro de la pantalla inicial donde se encuentra el inicio de sesión como un modal de registro. Se dispone de un formulario para el ingreso de los datos generales de registro, donde se escoge la función del empleado, y en caso de ser repartidores la Placa de su vehículo será un requisito obligatorio.

## Figura 40

### Formulario de registro de empleado



Nombre y Apellido...

Función de la empresa ▾

Número de Placa (Solo Repartidor)...

Correo Electrónico...

Contraseña...

Confirmar Contraseña...

*Nota: Modal con Formulario de registro de empleado. Elaborado por: Los autores.*

- **Registro de Productos**

- Para el registro se ha creado un formulario separado en 2 partes específicas. La parte superior carga la información principal del producto, mientras que en la parte inferior muestra los detalles generales de cada producto.
- Dentro de la información principal, el sistema permite cargar la imagen del producto que se está registrando. Para lograr este cometido se recurrió a la utilización de la librería **multer**. Esta librería es un middleware para el manejo multipart/form-data, que se usa principalmente para cargar archivos. Es necesario configurar el destino donde se guardará la imagen extraída, y el nombre del archivo a guardar que será en base a fecha actual en milisegundos para evitar duplicación de nombres.

## Figura 41

*Subida de archivos con Multer*

```
const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, "uploads/");
  },
  filename: function (req, file, cb) {
    cb("", Date.now() + "." + mimeTypes.extension(file.mimetype));
  },
});

const upload = multer({
  storage: storage,
});
```

*Nota: Configuración de Multer en app.js. Elaborado por: Los autores.*

- La imagen subida por el usuario podrá ser mostrada como una vista previa, mediante la utilización de acciones del Document Object Model (DOM), abstrayendo el id del componente, y cambiando el estado de la muestra de la imagen.

## Figura 42


*Imagen del Producto*

Registrar Producto

Nombre del Producto...

Proveedor...

Subir Imagen...



*Nota: Vista previo de imagen de un producto. Elaborado por: Los autores.*

- Al momento de guardar el formulario, el archivo es almacenado en el directorio local, y se abstrae el path que genera la imagen guardada para registrarla en la base de Datos, y pueda ser accedida posteriormente.

**Figura 43**  
*Abstracción del path del archivo*

```
const producto = {
  nombre: req.body.nombre,
  proveedor: req.body.proveedor,
  categoria: req.body.categoria,
  imagen: storage.filename,
  material: req.body.material,
  peso: req.body.peso,
  cm: req.body.cm,
  color: req.body.color,
  talla: req.body.talla,
  origen: req.body.origen,
  stock: req.body.stock,
  precioMer: req.body.precioMer,
  precioProv: req.body.precioProv,
};
const tempPath = req.file.path;
const targetPath = path.join(__dirname, tempPath);
```

*Nota: Obtención del path relativo del archivo. Elaborado por: Los autores.*

**Figura 44**  
*Path de la imagen del Producto*

aterial	color	stock	precio_mercado	precio_proveedor	foto
aracter varying (50)	character varying (50)	integer	real	real	character varying
astico	Rojo	2	35	20	C:\Users\fonsi\OneDrive - Universidad Politecnica Sal...
astico	Rojo	2	20	20	C:\Users\fonsi\OneDrive - Universidad Politecnica Sal...

*Nota: Path relativo de la imagen del producto en la base de datos. Elaborado por: Los autores.*

- **Registro de Clientes**

- Se creó un formulario simple creado a doble columna que sirve para colocar los datos más relevantes del usuario, siendo los elementos principales, su número de teléfono, y el medio de compra para poder tener un filtrado y distinción por medios de venta que tiene la venta. Para su construcción se utilizó HTML para su construcción. Y CSS y Bootstrap para su diseño.

**Figura 45**

*Formulario de registro del cliente*

Registrar Cliente

Nombre y Apellido... Edad...

Cédula... Teléfono...

E-mail... Medio de compra... ▼

Género... ▼

REGISTRAR

Admin

*Nota: Formulario de registro del cliente. Elaborado por: Los autores.*

- **Registro de Pedidos**

- Para poder registrar los pedidos fue necesario, que cada pedido esté vinculado a un usuario en específico, por lo que para registrar el pedido es necesario buscar el cliente.
- Para conseguir la validación, y un renderizado condicional se utilizaron helpers proveídos por la librería HandleBars, misma que permitió colocar una condición dentro del código HTML. Su sintaxis es `{{#if <variable a comparar>}}`, y se cierra con `{{/if}}`



## Figura 46

### Condicional HandleBars en HTML

```
</form>
<hr> {{#if cliente}}
<div class="row">
  <div class="col-5">
    <div class="card border-left-info shadow h-100">
      <div class="card-body">
        <div class="row no-gutters align-items-center">
          <div class="col mr-2">
            <div class="text-xs font-weight-bold text-info text-
            </div>
          <div class="row no-gutters align-items-center">
            <div class="col-auto">
```

*Nota: Condicional de HandleBars dentro de una plantilla HTML. Elaborado por: Los autores.*

- Una vez validada la existencia del usuario, se carga un formulario que permite escoger los pedidos, cargando la lista de los productos disponibles que pueden ser agregados, además validando que la cantidad escogida no supere al número de existencias de cada producto.
- Para la carga de productos y variables se utilizó la función de HandleBars **{{#each}}**, y se cierra con **{{/each}}**, donde para acceder a cada variable obtenido del bucle, se utiliza la palabra clave **this**.

## Figura 47

Plantilla HTML de la carga de productos

```
<table class="table table-bordered" id="dataTable" width="100%" cellspacing="0">
  <thead>
    <tr>
      <th class="text-center">Nombre Producto</th>
      <th class="text-center">Cantidad</th>
      <th class="text-center">Total Producto</th>
      <th class="text-center">Subtotal</th>
      <th class="text-center">Opción</th>
    </tr>
  </thead>
  <tbody>
    {{#each productos}}
    <tr>
      <th class="text-center">{{this.nombreProd}}</th>
      <th class="text-center">{{this.cant}}</th>
      <th class="text-center">{{this.total}}</th>
      <th class="text-center">{{this.subtotal}}</th>
      <th class="text-center"><form action="/quitarProd" method="post">
        <input type="text" name="quitar" value="{{this.codigoProd}}" style="display:none;">
        <button type="submit" class="btn btn-danger btn-sm">quitar</button>
      </form></th>
    </tr>
    {{/each}}
  </tbody>
</table>
```

Nota: Plantilla HTML de la tabla de productos con HandleBars. Elaborado por: Los autores.

## Figura 48

Interfaz del Registro del Pedido

Registro del Pedido [Nuevo Pedido](#)

Cédula Cliente...

**DATOS DEL CLIENTE:**  
Cédula: 1705224796  
Nombre: María Moya  
Edad: 61  
Teléfono: 0985519588  
Medio de Compra: WhatsApp

Producto...  Cant

**Resumen del Pedido**

Nombre Producto	Cantidad	Total Producto	Subtotal	Opción
Victoria Secret Daisy Haze	1	20	20	<input type="button" value="quitar"/>
Zapatos Converse	1	20	20	<input type="button" value="quitar"/>

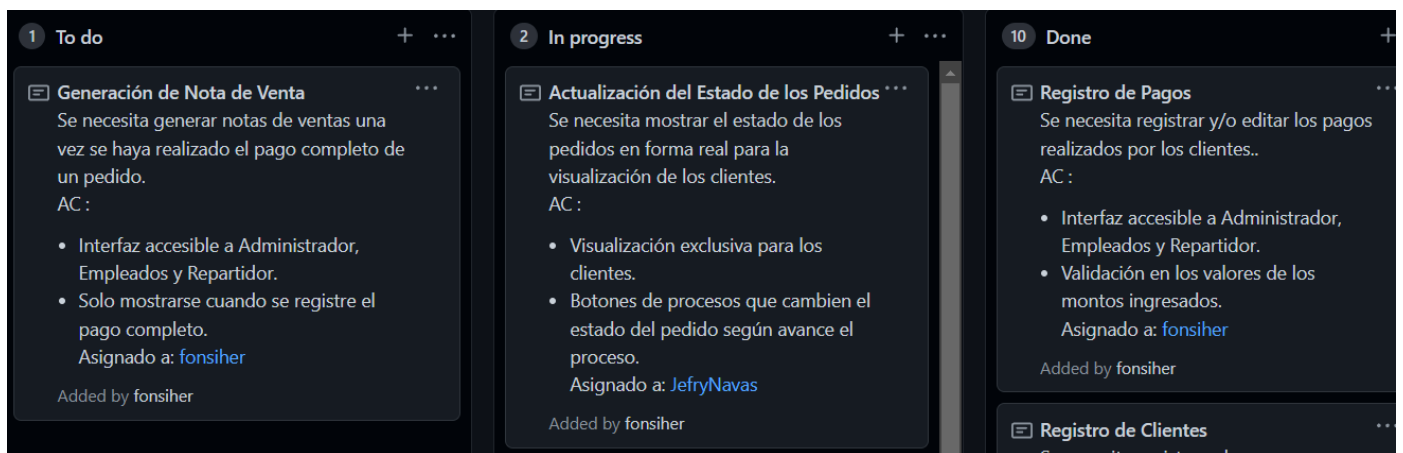
Nota: Interfaz del registro de un pedido. Elaborado por: Los autores.

#### 4.1.3.4. *Sprint 4: Proceso de Pago y Seguimiento de Pedido*

**Objetivo del Sprint:** Generar tanto el aspecto visual como la lógica necesaria para el manejo del proceso de pagos y seguimiento de pedidos, mediante la interconexión secuencial de interfaces, y técnicas de actualización de estado, con el fin de conseguir que el sistema pueda registrar los pagos, y lograr que los clientes y empleados tengan un seguimiento del proceso que la tienda realiza con cada pedido.

#### **Figura 49**

*Tablero Kanban durante el Sprint 4*



*Nota: Tablero Kanban durante el Sprint 4. Elaborado por: Los autores.*

- **Registro de Pagos**
  - El registro de los pagos está concatenado por un botón que puede ser accesible por los empleados, repartidores y el administrador del sistema. El panel de pago mostrado a continuación es mostrado únicamente al momento de la entrega del pedido. El formulario permite ingresar la información del pago, vinculado con el código de cada pedido.

**Figura 50**  
*Formulario de Registro de pago*

Registrar Pagos

Pedido: pedido-1626587925478

Saldo a Pagar (\$):

30

Efectivo

30

Registrar Pago

*Nota: Formulario para el registro de pago de un pedido. Elaborado por: Los autores.*

- Para los casos en los que el pago inicial no cubra el costo completo del pedido. El sistema cuenta con acceso a los pedidos pendientes de pago, mismos que muestran un botón que redirecciona a un panel similar al mostrado anteriormente, indicando el saldo pendiente del cliente.
- En primer lugar, el sistema calcula según el pedido el saldo a pagar comparando el pago pendiente, y accediendo mediante un bucle al precio total del pedido, con datos provenientes desde la Base de Datos.

**Figura 51**  
*Cálculo del saldo pendiente*

```
let resumen = await buscarPorPedido(codigo);  
let a_pagar = 0;  
resumen.forEach((element) => {  
  a_pagar += element.total;  
});  
let saldo = resumen[0].saldo;
```

*Nota: Código para obtener el saldo pendiente de un pedido. Elaborado por: Los autores.*

- Con cada registro de pago se evalúa el pago realizado con el saldo pendiente, mediante una resta simple, seguido de una evaluación condicional para establecer el nuevo estado del pedido según el saldo pendiente.

### Figura 52

*Verificación del saldo de un pedido*

```
let saldo = total - monto;|
let rp = "";
let estado = "";
if (saldo > 1) {
  rp = `Cuota Registrada, Saldo Pendiente de: ${saldo} `;
  estado = "Pago Parcial";
} else {
  rp = "Pago Registrado";
  estado = "Cancelado";
}
```


*Nota: Código para verificar el saldo y estado del pago del pedido. Elaborado por: Los autores.*

- **Generación de Nota de Venta**

- El panel es accesible por los empleados, repartidores y el administrador del sistema. Se muestra de manera automática una vez el pago del pedido sea completado sin ningún saldo remanente.
- Se despliega un formulario cargado por defecto con los datos del cliente que realizó el pedido, cargando la fecha de generación de la nota de venta, y el detalle del pedido realizado. Una vez verificados los datos, se guardan en la base de datos para ser visibles tanto para el cliente, como para el administrador de la empresa.

## Figura 53

### Formulario Registro de la Nota de Venta

Admin 

#### Datos de Facturación

<input type="text" value="13/12/2022"/>	<input type="text" value="0985519588"/>
<input type="text" value="María Moya"/>	<input type="text" value="Dirección..."/>
<input type="text" value="1705224796"/>	Subtotal: <input type="text" value="26,4"/>
<input type="text" value="marimoya2112@hotmail.com"/>	Total Pago: <input type="text" value="30"/>

**Detalles del Pedido**

Nombre Producto	Cantidad	Total Producto
Victoria Secret Daisy Haze	2	40
Boxers Tommy Hilfiger	2	30

*Nota: Formulario para el registro de la Nota de Venta. Elaborado por: Los autores.*

- **Actualización de Estado de los Pedidos**

- Para el funcionamiento de este requerimiento se abstrajeron las acciones que van realizando los diversos funcionarios de la empresa, ejecutando cambios del estado del pedido en la base de Datos, mismo que es mostrada en el perfil de los clientes por cada pedido que realicen.
- El estado del pedido empieza a registrarse una vez se guarde un nuevo pedido. El código para manejar las acciones indicadas anteriormente despliega una consulta de actualización hacia la base de datos, de una manera directa gracias al uso de la librería pg.

## Figura 54

### Consulta actualización del pedido

```
const updateEstado = async(codigo_ped, metodo) => {
  try {
    const consulta = `update pedido set estado = 'Entregado-${metodo}' where cod_ped = '${codigo_ped}'`;
    const res = await pool.query(consulta);
    if (res.rowCount == 1) {
      return "Estado Actualizado";
    } else
      return "No se pudo actualizar el estado";
  } catch (error) {
    return error.message;
  }
}
```

*Nota: Consulta de actualización del estado del pedido. Elaborado por: Los autores.*

- Para la visualización de los datos del pedido, se recurrió a diversos métodos que extraen los valores de la Base de Datos, tal como el precio final a pagar, posibles pagos pendientes, e información del conductor que realizará la entrega del pedido.
- Al tener que llamar varias consultas conjuntas, es esencial aplicar en diversas funciones el concepto de asincronismo. El asincronismo en Javascript se lo logra de varias maneras, donde a lo largo de este proyecto se recurrió al uso de las funciones asíncronas *async* – *await*, con el objetivo de tener una lectura más comprensible de código, y mejor manejo de las funciones invocadas. (Gokhale et al., 2021)

## Figura 55

### Función asíncrona Seguimiento del pedido

```
const seguimientoPedido = async(req, res) => {
  if (req.session.loggedinCliente) {
    let user = req.session.user;
    const codigo = req.body.id;
    let resumen = await buscarPorPedidoSS(codigo);
    let cond = resumen[0].conductor;
    let datoscond = await getDatosConductor(cond);
    let a_pagar = 0;
    resumen.forEach((element) => {
      a_pagar += element.total;
    });
  }
};
```

*Nota: Función asíncrona para obtener los datos del pedido. Elaborado por: Los autores.*

- Con los datos abstraídos, la interfaz permite al cliente acceder a la información de cada uno de sus pedidos realizados, tanto pasados como actuales. Cuenta con 4 estados que va cambiando de acuerdo con el estado del pedido que devuelve la Base de Datos.
- La interfaz generada muestra la información más importante, incluyendo una estimación de la fecha de entrega, y los datos reales de la persona encargada de entregar el pedido.

## Figura 56

### Interfaz Seguimiento del Pedido

PEDIDO: [pedido-1626823040994](#) Fecha esperada de entrega: 2021-07-20  
Total del Pedido: \$110

Orden Procesada    Orden Asiganda    Orden En Ruta    Orden Arribada

Repartidor Asignado: **Fernando Hierro** Placa: **AH241C**

*Nota: Interfaz del cliente para el seguimiento del pedido. Elaborado por: Los autores.*



- Para conseguir el efecto presentado en la figura anterior, se utilizan evaluaciones de Handlebars, en conjunto con cambios ejecutados a través de acciones del DOM, y código en CSS.
- Mediante condicionales se evalúa el estado en el que se encuentra el pedido, según el estado obtenido, se va modificando el identificador de las clases, que son leídas y modificadas mediante CSS, para lograr que se pinte de forma secuencial el estado en el que se encuentra el pedido.

### Figura 57

*Evaluaciones del Estado en la Interfaz*

```

<div class="col-12">
  {{#if (eq estado "Pedido Registrado")}}
  <ul id="progressbar" class="text-center">
    <li class="active step0"></li>
    <li class="step0"></li>
    <li class="step0"></li>
    <li class="step0"></li>
  </ul>
  {{/if}}
  {{#if (eq estado "Conductor Asignado")}}
  <ul id="progressbar" class="text-center">
    <li class="active step0"></li>
    <li class="active step0"></li>
    <li class="step0"></li>
    <li class="step0"></li>
  </ul>
  {{/if}}
  {{#if (eq estado "Pedido en camino")}}

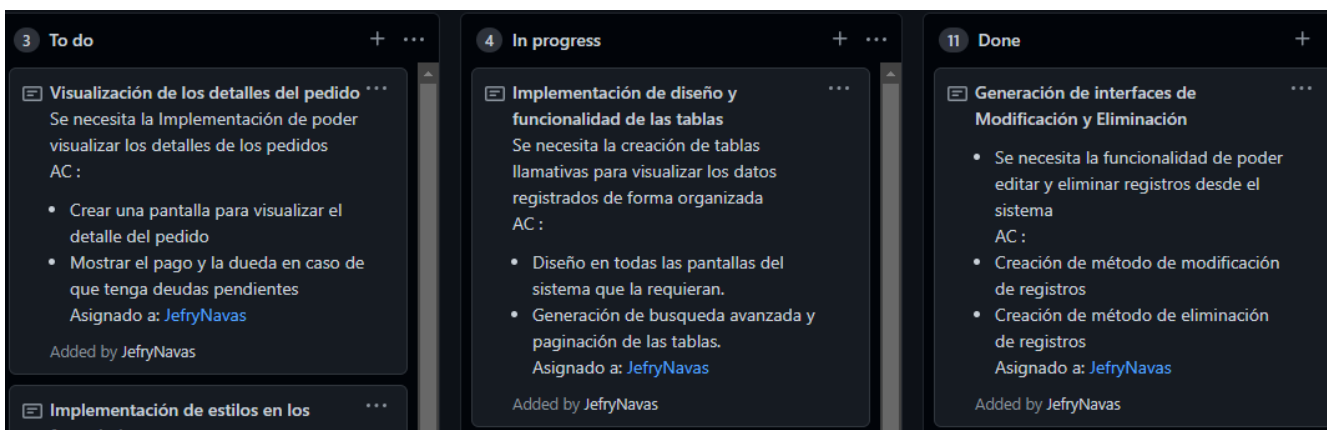
```

*Nota: Evaluaciones para mostrar el estado en la interfaz. Elaborado por: Los autores.*

#### 4.1.3.5. *Sprint 5: Tablas de Administración y Paneles Informativos para los Usuarios*

**Objetivo del Sprint:** Implementar diversas tablas y paneles de visualización de los datos guardados en la Base de Datos de la tienda tanto para los empleados como clientes, agregando acciones especiales de modificación para el administrador, mediante generación de peticiones HTTP, y filtros específicos, para que la aplicación muestre de una forma organizada y atractiva a la vista los datos registrados en la Base de Datos.

**Figura 58**  
*Tablero Kanban durante el sprint 5*



*Nota: Tablero Kanban durante el sprint 5. Elaborado por: Los autores.*

En cuanto al diseño de la interfaz, cuenta con varias interfaces, como las pantallas de tablas de productos, usuarios, ganancias de repartidores. También se han utilizado algunos elementos de Bootstrap para hacerlo más agradable estéticamente.

Para la funcionalidad como búsqueda avanzada, paginación de las tablas se utilizó dataTables que es una librería de jQuery y Bootstrap programado en JavaScript.

**Figura 59**  
*Tabla de Productos*

Tabla de Productos

Show  entries

Search:

Nombre	Categoría	Proveedor	Color	# en Stock	Precio Venta	Precio Proveedor	Modificar	Eliminar
Boxers Tommy Hilfiger	Ropa	Lola Larco	Varios	1	20	7		
Buzo Puma	Ropa	Lola Larco	Gris	3	50	35		
Camiseta Puma	Ropa	Paulina Mora	Blanco	1	30	20		
Hoddie Nike	Ropa	Paulina Mora	Negro	2	60	45		

*Nota: Interfaz de la tabla de los pedidos. Elaborado por: Los autores.*

**Figura 60**  
*Tabla de Usuarios*

Tabla de Usuarios

Show  entries

Search:

Id Usuario	Tipo Usuario	Nombre	Email	Modificar	Eliminar
1	Administrador	Admin	admin@invirtual.com		
3	Repartidor	Andrés Quintana	andresq@invirtual.com		
4	Empleado	Raquel Ripalda	raquelripalda@gmail.com		
5	Repartidor	Fernando Hierro	fhierro@invirtual.com		
7	Repartidor	Henry Hurtado	henryh@invirtual.com		

Showing 1 to 5 of 5 entries

Previous **1** Next

*Nota: Código para verificar el saldo y estado del pago del pedido. Elaborado por: Los autores.*

## Figura 61

### Tabla de Repartidores y sus Ganancias

Repartidores y sus Ganancias			
Show	Search:		
10	<input type="text"/>		
entries			
Nombre del Repartidor	Email	Fecha	Ganancia Total del Día
Andrés Quintana	andresq@invritual.com	Wed Jul 07 2021 00:00:00 GMT-0500 (hora de Ecuador)	10
Andrés Quintana	andresq@invritual.com	Sat Jul 10 2021 00:00:00 GMT-0500 (hora de Ecuador)	3
Andrés Quintana	andresq@invritual.com	Wed Jul 14 2021 00:00:00 GMT-0500 (hora de Ecuador)	8
Andrés Quintana	andresq@invritual.com	Sat Jul 17 2021 00:00:00 GMT-0500 (hora de Ecuador)	5
Andrés Quintana	andresq@invritual.com	Sun Jul 18 2021 00:00:00 GMT-0500 (hora de Ecuador)	5
Fernando Hierro	fhierro@invritual.com	Wed Jul 07 2021 00:00:00 GMT-0500 (hora de Ecuador)	3

*Nota: Tabla de repartidores y sus ganancias. Elaborado por: Los autores.*






Para la visualización del detalle de los pedidos, el administrador cuando accede a la opción “*todos los pedidos*”, por entregar, asignados o pendientes de pago puede visualizar los datos del cliente junto con el detalle del pedido y el costo total.

**Figura 62**  
*Tabla de Pedidos*

Todos los Pedidos

Show:  entries

Search:

Código Pedido	Fecha de Entrega	Dirección	Estado	Detalles
pedido-1665442945736	2022-10-10	Catarama y Zumbagua	Pedido Registrado	
pedido-1665873235798	2022-10-15	Catarama y asd	Pedido Registrado	
pedido-1670985189633	2022-12-13	Catarama y Zumbagua	Pedido Registrado	
pedido-1670985281459	2022-12-13	Amaluza y Jose Peralta	Pedido Registrado	
pedido-1670985452876	2022-12-13	Alvarado y Francia	Pedido Registrado	

Showing 1 to 5 of 5 entries (filtered from 22 total entries)

Previous **1** Next

*Nota: Tabla de todos los pedidos. Elaborado por: Los autores.*

**Figura 63**  
*Detalle del pedido*

Cliente: Jorge Mendez  
 Teléfono: 0996434838

Detalles del Pedido: pedido-1626659476035

Show:  entries

Search:

Producto	Cantidad	Total Producto
Boxers Tommy Hilfiger	1	15
Victoria Secret Daisy Haze	1	20

Showing 1 to 2 of 2 entries

Previous **1** Next

**Costo total del Pedido: \$35**

*Nota: Detalle del pedido registrado. Elaborado por: Los autores.*

Para la modificación de los formularios, el administrador es el único que puede acceder a estas pantallas ya que son exclusivas para la administración de la tienda a continuación se presenta algunas pantallas para modificar los formularios.

**Figura 64**  
*Interfaz Modificar Producto*

Modificar

Nombre:

---

Detalles Generales

Material:  Color:

Peso:  Talla:

Tamaño:  Origen:

Stock:

---

Precio Mercado:  Precio Proveedor:

*Nota: Interfaz para la modificación del producto. Elaborado por: Los autores.*

**Figura 65**  
*Interfaz Modificar Usuario*

Modificar

Admin

Administrador

admin@invirtual.com

.....

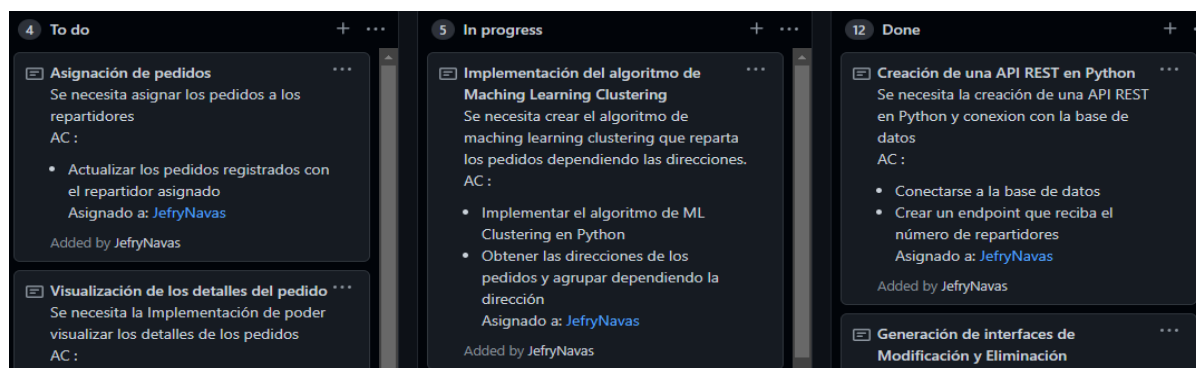
Modificar Usuario

*Nota: Interfaz para la modificación de un usuario. Elaborado por: Los autores.*

#### **4.1.3.6. Sprint 6: Gestión de la Repartición de Pedidos**

**Objetivo del Sprint:** Configurar la lógica de programación con Python, en conjunto con las interfaces destinadas a gestionar la asignación de pedidos a los repartidores de la tienda mediante la técnica de Machine Learning “Clustering”; para lograr una organización inteligente y optimizada que deriva en un ahorro significativo de tiempo en los pedidos que son repartidos por la tienda día a día.

**Figura 66**  
*Tablero Kanban durante el sprint 6*



*Nota: Tablero Kanban durante el sprint 6. Elaborado por: Los autores.*

Para realizar la funcionalidad de asignación de los pedidos se creó un servicio API REST en el lenguaje de programación Python con el framework Flask el cual nos facilita la creación de las aplicaciones web.

En el siguiente fragmento de código se expone un endpoint que recibirá como argumento el número de los repartidores desde el sistema. Para lo cual el servicio se deberá conectar a la base de datos y obtener los pedidos en estado registrado del día actual.

**Figura 67**  
*Interfaz Asignar Repartidores del día*

Asignar Repartidores a los pedidos del día de hoy: 2022-12-13

2

Lista de Pedidos:

Código Pedido	Fecha de Entrega	Estado	Detalles
pedido-1670985452876	2022-12-13	Pedido Registrado	
pedido-1670985189633	2022-12-13	Pedido Registrado	
pedido-1670985281459	2022-12-13	Pedido Registrado	

*Nota: Interfaz de Asignación de los repartidores. Elaborado por: Los autores.*

Mediante la técnica de Machine Learning “Clustering”, que se implementó, se pudo lograr repartir los pedidos dependiendo las direcciones que se han registrado, retornando como json un mensaje de “PEDIDOS DE HOY” con la información de asignación de los repartidores.

Como validación al momento de llamar al endpoint si no existen pedidos registrados en el día responde un json con el mensaje “NO HAY PEDIDOS POR ASIGNAR”. A continuación, se



muestra el fragmento de código del endpoint que devuelve los pedidos en formato json con los repartidores asignados.

**Figura 68**  
*API Clustering*

```
@app.route('/ml/<int:cluster>')
def getCluster(cluster):
    repartidores = getDirecciones()
    direcciones = []
    for ubi in getDirecciones():
        direccion = ubi['direccion']
        direcciones.append(direccion)
    lon = len(getDirecciones())
    data = geocoders(direcciones,cluster)
    if data:
        for k in range(lon):
            nums = []
            for i in data:
                num = int(i)
                nums.append(num)
            repartidores[k]['Repartidor'] = nums[k]
        return jsonify({'Repartidores': {
            'estado': True,
            'message': 'PEDIDOS DE HOY',
            'data': repartidores}})
    else:
        return jsonify({'Repartidores': {
            'estado': False,
            'message': 'NO HAY PEDIDOS POR ASIGNAR'}})
```

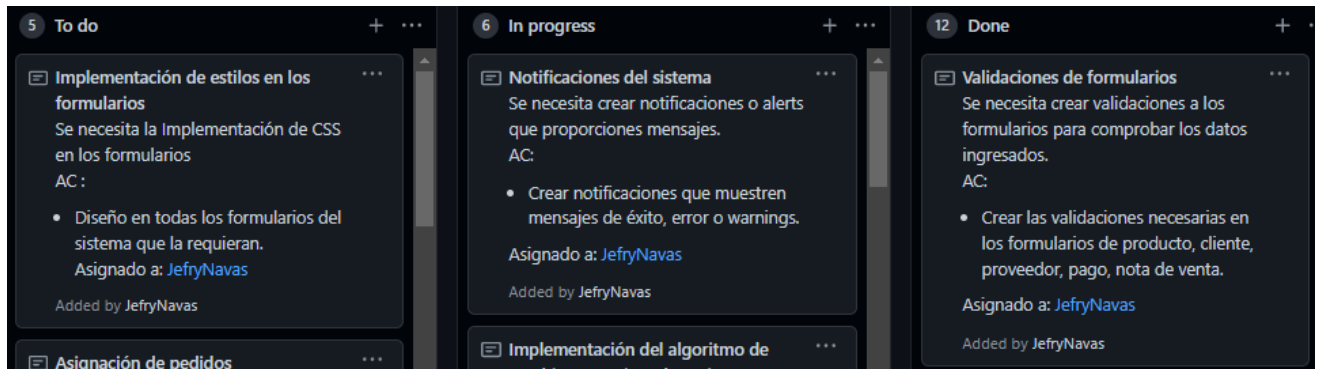
*Nota: Código para generar la repartición de los pedidos. Elaborado por: Los autores.*

#### 4.1.3.7. *Sprint 7: Validaciones Finales y Comprobaciones.*

**Objetivo del Sprint:** Realizar las diferentes validaciones en cada una de las interfaces y paneles creados que se muestran dentro de la aplicación, mediante la utilización de librerías especiales de validación y comprobación constante de los datos requeridos, con el fin de lograr abstraer dentro de la misma ejecución de la aplicación los posibles errores que puedan cometer los usuarios sin que la aplicación detenga su funcionamiento.

## Figura 69

Tablero Kanban durante el sprint 7



*Nota: Tablero Kanban durante el sprint 7. Elaborado por: Los autores.*

Para las notificaciones de utiliza la librería sweetalert2 la cual nos sirve para crear cuadros emergentes de JavaScript con un diseño profesional y fácil de personalizar e implementar.

## Figura 70

Alertas del Sistema



Producto Modificado  
Correctamente

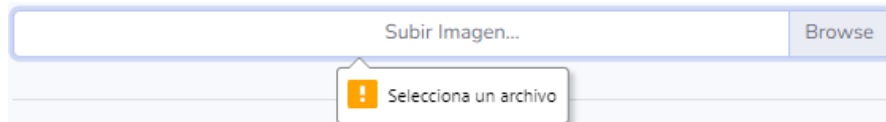
*Nota: Mensajes de alertas en el sistema. Elaborado por: Los autores.*

Para validar los diferentes formularios, se utilizaron las validaciones proveídas directamente desde las etiquetas HTML.

## Figura 71

Validaciones Formularios en el Sistema

The image shows a form with two input fields. The first field is labeled 'Nombre y Apellido...' and is empty. The second field is labeled 'Cédula...' and has a validation error message: 'Completa este campo'.



*Nota: Mensajes de validación en los formularios. Elaborado por: Los autores.*

## 3.2 PRUEBAS

En el siguiente subcapítulo se realizaron pruebas de código y pruebas de carga a distintas rutas de la aplicación web de InVirtual Store, con el fin de comprobar la calidad del código, y el funcionamiento correcto de la aplicación bajo distintas cargas.

### 3.2.1 Pruebas del Código

Las pruebas de código permiten obtener métricas sobre el código creado, identificando posibles fallos y puntos sensibles que pueden ir mejorando, con la finalidad de que el código sea más limpio y seguro.

Para ejecutar las pruebas de código, se recurrió a la plataforma de código abierto SonarQube. Esta aplicación realiza un análisis de código estático, buscando errores de código, calidad del código mediante reglas de codificación, duplicaciones de código, y la arquitectura seguida. Además, todo este análisis es abstraído y mostrado en paneles de control detallados que muestran los resultados de la prueba. (Gigleux & Campbell, 2016)

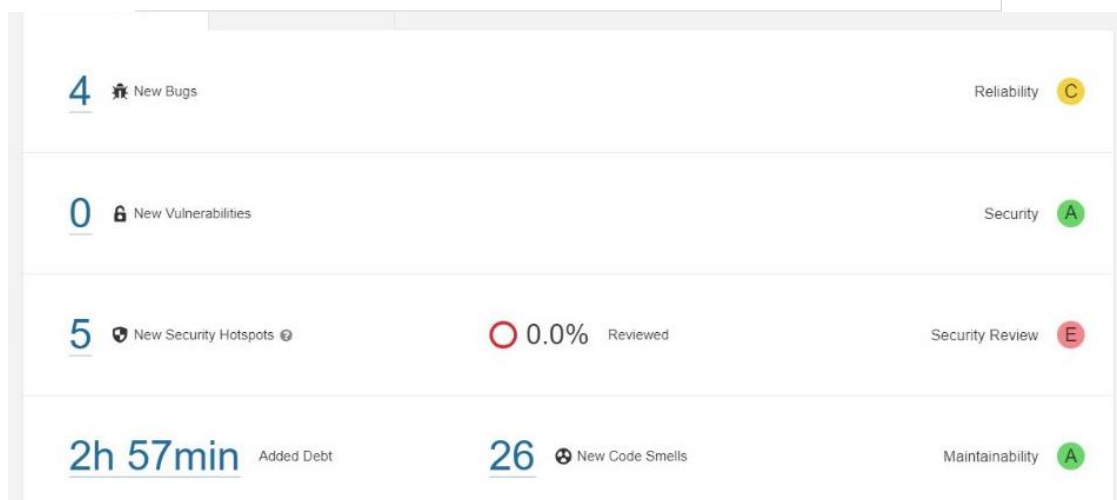
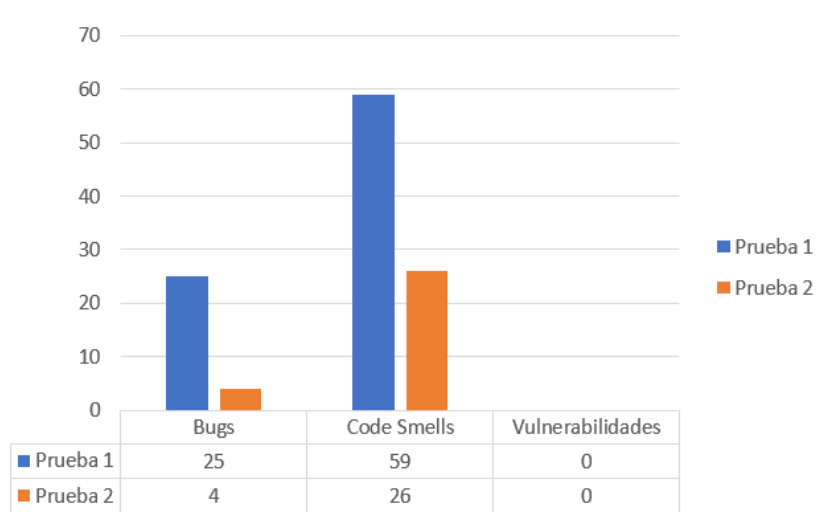
Para vincular el código con la aplicación se agregó el archivo de configuración: ***sonar-project.properties*** dentro del directorio de la aplicación, especificando que el análisis se centre en los archivos dentro de la carpeta **src**, priorizando a los archivos de JavaScript con extensión **.js**, y excluya archivos de estilos y de configuración.

SonarQube en su primer análisis mostró que dentro del código fuente analizado se encontraron 25 bugs, 59 code smells, 7 posibles puntos de problemas de seguridad y 0 vulnerabilidades.

Una vez ejecutados los arreglos, se ejecutó un segundo análisis que demostró una disminución significativa de los problemas en el código devolviendo, 4 posibles bugs, 2 posibles puntos de problemas de seguridad, y 26 Code Smells, tal y como se puede observar en la Figura 71.

**Figura 72**

*Gráfico de Resultado de Pruebas de código con SonarQube*



*Nota: Gráfico de problemas detectados en el código en un intervalo de tiempo. Elaborado por:  
Los autores*

Respecto a los bugs que se encontraron en la aplicación, marcan problemas por posibles efectos secundarios al sistema. Estos errores comúnmente son causados por las variables establecidas al momento de iniciar sesión el sistema. SonarQube sugirió manejar a las variables de sesión como una función y más no, como un objeto llamado de forma independiente.

Los bugs se encontraron principalmente en las funciones, que funcionan como controladores de los servicios de la aplicación, mostrando un tiempo de esfuerzo de 10 minutos para resolverlos. Los cambios fueron realizados en los archivos, donde los cambios no representaban un cambio drástico en el funcionamiento del código, logrando solventar un total de 21 bugs.

Respecto a los puntos de seguridad vulnerables, el análisis encontró 3 categorías en las que la seguridad de la aplicación puede verse afectada. DoS (Denial of Service), criptografía débil y huella de la tecnología web.

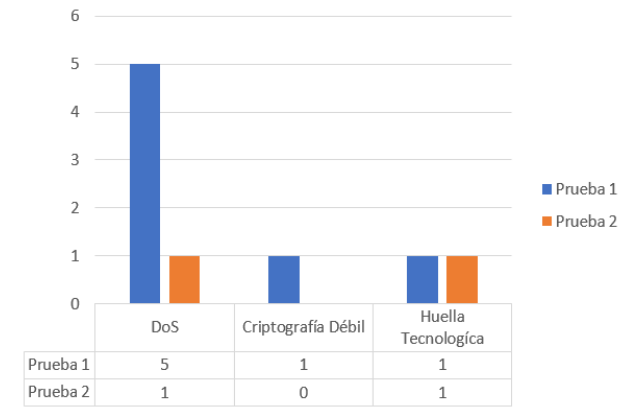
La mayoría de los puntos encontrados se encuentran en la categoría de DoS, donde se procedió a reparar los problemas que fueron encontrados, logrando una disminución en los problemas de seguridad dentro de la aplicación, tal y como se muestra en la Figura 72.

Los problemas de DoS se encontraron principalmente en los archivos que utilizan funciones extraídas desde la librería multer. Esta librería es necesaria para la carga de archivos de los usuarios al servidor, misma que sin un límite de espacio establecido, la aplicación es susceptible a un ataque

de DoS, provocando una saturación de los puertos del servidor sobrecargándolo, y ocasionando que la aplicación esté obligada a interrumpir su operación. (Eliyan & di Pietro, 2021)

**Figura 73**

*Problemas de seguridad por Categoría y Prueba Ejecutada*



*Nota: Gráfico de análisis de Problemas de Seguridad separado por categorías. Elaborado por: Los autores.*

Los Code smells son regularmente referidos a malas prácticas ejecutadas en la realización del código, posibles puntos donde la lógica, arquitectura o sintaxis utilizada puede ser mejorada con el objetivo de obtener un código más comprensible, y en algunos casos mejorar su rendimiento.

Las recomendaciones señaladas respecto a los Code smells están basadas en una eliminación de impresiones de consola no detectadas, y refactorización del código en condiciones buscando evitar comparaciones innecesarias, que ya son sobreentendidas por el lenguaje de programación.

**3.2.1.1 Pruebas de Rendimiento**

InVirtual Store al ser un emprendimiento en crecimiento, cuenta con un reducido número de empleados, y la lista de clientes diarios también es baja. Actualmente cuentan con un promedio de

máximo 5 pedidos diarios, con miras a seguir creciendo. Tomando esto en cuenta, es necesario probar el rendimiento del sistema, evaluando escenarios de una recurrencia alta de usuarios, de acuerdo su número actual de empleados y promedio de clientela diaria.

Con el objetivo de obtener métricas de rendimiento más precisas, las pruebas son realizadas a la aplicación que se encuentra desplegada en el servidor de producción. Para la ejecución de las pruebas se utilizó el servicio en línea de pago, Azure Load Testing.

Azure Load Testing permite que los desarrolladores generen simulaciones de carga y ejecución a gran escala que revelen información procesable sobre el rendimiento de la aplicación, la escalabilidad y la capacidad de la aplicación, con el objetivo de encontrar cuellos de botella por una carga alta de peticiones, en busca de brindar información para optimizar la escalabilidad o capacidad de la aplicación si es necesario. (Trogh et al., 2022)

Las pruebas de rendimiento se realizaron en 3 rutas específicas de la aplicación que son las más utilizadas tanto por empleados como por clientes de la tienda. Para los empleados se utilizó un escenario de 10 usuarios, accediendo cada 2 segundos durante el transcurso ininterrumpido de 2 minutos. Mientras que, para los clientes se probó un escenario similar al de los empleados, tan solo aumentando el número de usuarios a 15.

Cada prueba devuelve estadísticas de rendimiento arrojados por la aplicación, mostrando el número total de solicitudes ejecutadas, el tiempo promedio de respuesta, el porcentaje de errores y el rendimiento de la aplicación según las solicitudes realizadas al servidor.

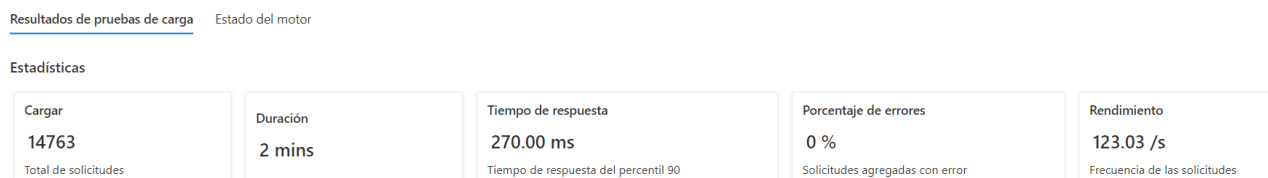
### 3.2.2.1. Rendimiento en la pestaña creación de Pedidos.

Se evaluó el rendimiento en la ruta **/pedidos** de la aplicación, a esta ruta tienen acceso el administrador y los empleados del sistema. Esta ruta es de las más utilizadas, ya que sirve para registrar los nuevos pedidos en el sistema.

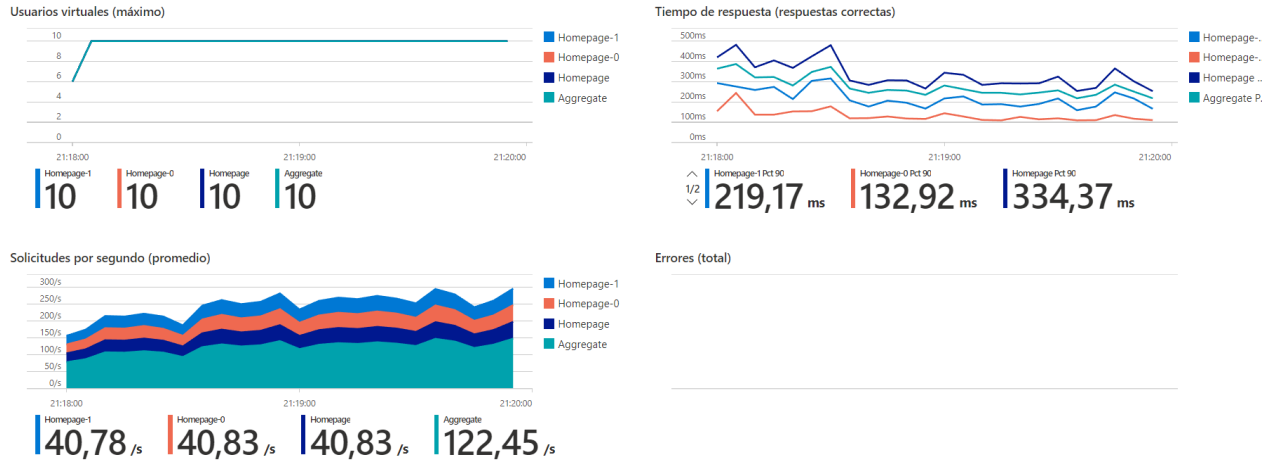
Los gráficos de rendimiento de las pruebas indican el tiempo de carga que toma el servidor para resolver cada petición ejecutada para los 10 usuarios establecidos. Se resalta el agregado total del tiempo promedio que tiene el servidor en resolver las peticiones.

Se ha mostrado un rendimiento similar en cada petición, con una resolución rápida y efectiva, menor a los 0.5 segundos, indicando un nivel constante con pequeños picos en ciertas peticiones que superan los 400 ms en el procesamiento de cada petición. Además, muestra una gran fiabilidad en las repuestas del servidor ya que devuelve un nulo porcentaje de errores indicando que todas las peticiones han podido ser procesadas correctamente.

**Figura 74**  
*Reporte de rendimiento de la ruta /pedidos*







*Nota: Gráfico del rendimiento de la ruta /pedidos. Fuente: Azure Load Testing*

### 3.2.2.2. Rendimiento en la pestaña de Lista de Pedidos Asignados

Se evaluó el rendimiento en la ruta **/tablaAsignados**, a esta ruta tienen acceso el administrador, los empleados de la tienda, y los repartidores. Es una de las rutas más utilizadas en la aplicación ya que muestra la lista de pedidos listos y asignados para su entrega con toda la información del cliente, así como el resumen del pedido.

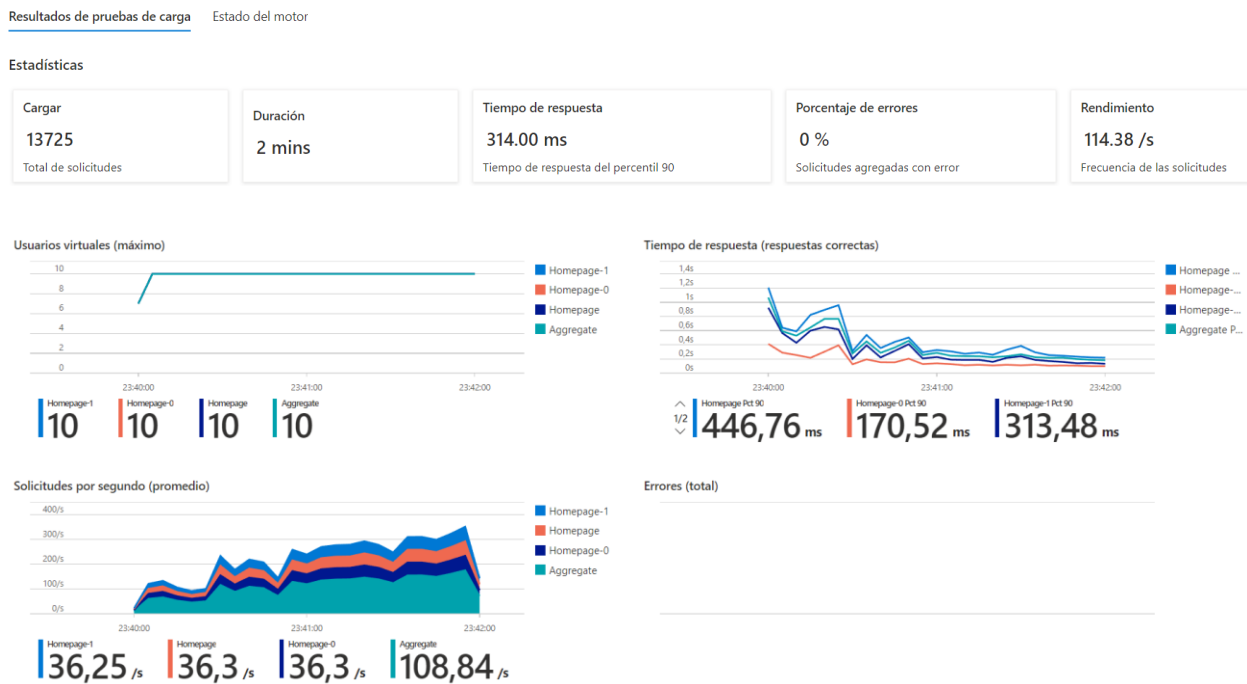
Los gráficos de rendimiento de las pruebas indican el tiempo de carga que toma el servidor para resolver cada petición ejecutada para los 10 usuarios establecidos, donde analiza principalmente el promedio de la resolución de cada petición enviada al servidor.

Las pruebas mostraron un rendimiento eficiente, donde el tiempo de respuesta fue disminuyendo con el transcurso de las peticiones debido a las acciones de caché ejecutadas, mostrando unos picos iniciales superiores 1 segundo, y a continuación fue disminuyendo progresivamente hasta llegar a

un mínimo de peticiones resueltas en 0.2 ms, tan solo mostrando unos pequeños picos intermedios ligeramente superiores a un tiempo de respuesta de 0.4 ms.

De igual manera, la ruta mostró una gran fiabilidad, ya que cuenta con un nulo porcentaje de errores indicando que todas las peticiones han podido ser procesadas correctamente.

**Figura 75**  
*Reporte de rendimiento de la ruta /tablaAsignados*



*Nota: Gráfico del rendimiento de la ruta /tablaAsignados. Fuente: Azure Load Testing*

### 3.2.2.3. Rendimiento en la pestaña de Lista de Pedidos (Cliente)

Se evaluó el rendimiento en la ruta **/tusPedidos**, esta ruta es accesible únicamente para los clientes InVirtual Store. Esta pestaña es la más utilizada por los clientes en la aplicación, ya que muestra

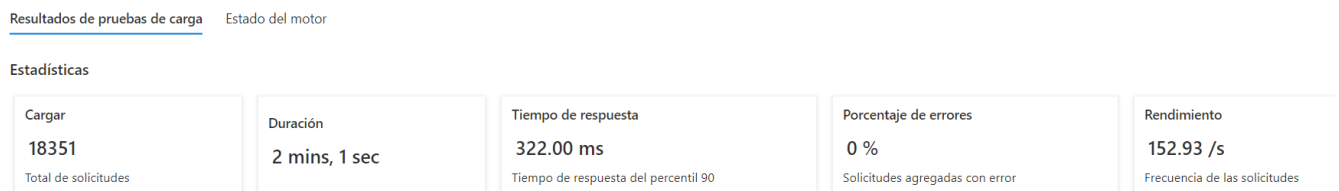
la lista de todos los pedidos realizados por cada cliente históricamente, indicando el estado, y la información más importante de cada pedido.

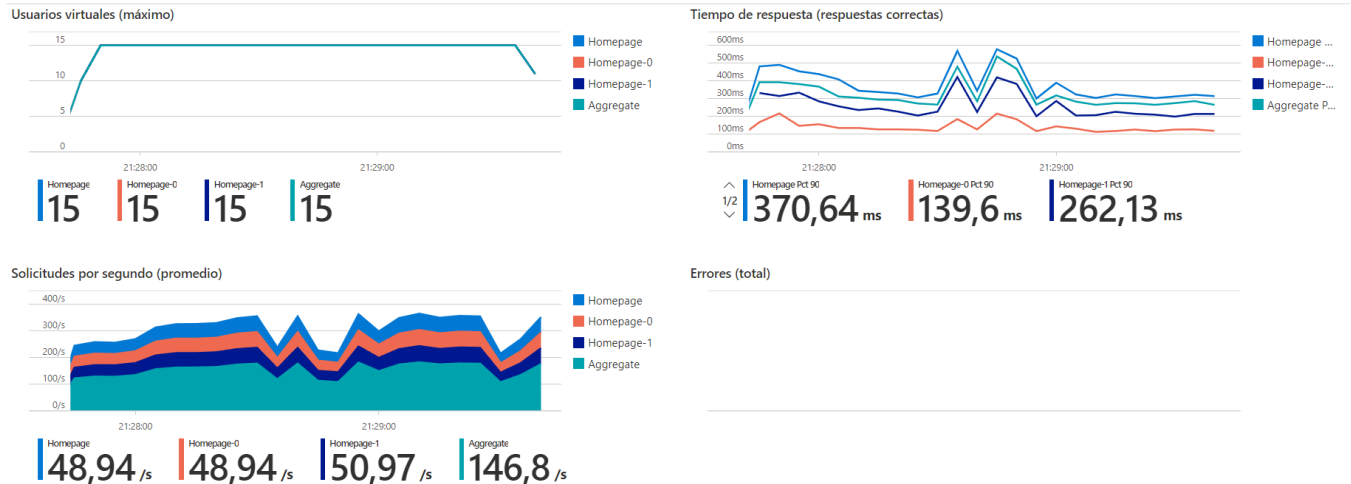
Los gráficos de rendimiento de las pruebas indican el tiempo de carga que toma el servidor para resolver cada petición ejecutada para los 15 clientes establecidos.

Las pruebas mostraron un rendimiento eficiente, pero con un comportamiento variado el tiempo de respuesta. Inicialmente se mantuvo con un rendimiento similar que rondaba entre los 300 a 400 ms, pero en la mitad de la prueba, el tiempo de respuesta mostró oscilaciones con picos muy altos cercanos a los 500 ms, que bajaron drásticamente. Para el final del tiempo de la prueba, las oscilaciones se estabilizaron y se mantuvo un nivel de respuesta casi constante con un tiempo de respuesta muy similar al devuelto al inicio de la prueba.

Adicionalmente, la ruta cuenta con una gran fiabilidad, ya que muestra un nulo porcentaje de errores en la resolución de peticiones, es decir que cada una de las peticiones ejecutadas fueron procesadas correctamente.

**Figura 76**  
*Reporte de rendimiento de la ruta /tusPedidos*





*Nota: Gráfico del rendimiento de la ruta /tusPedidos. Fuente: Azure Load Testing*

## 4. IMPLEMENTACIÓN

El siguiente subcapítulo mostrará el proceso de implementación de la aplicación, para ser desplegada en servidores, mediante la utilización de servicios de Microsoft Azure en conjunto con GitHub. Se presentan las configuraciones realizadas para desplegar la aplicación, al igual que el diagrama de despliegue.

### 4.1 DIAGRAMA DE DESPLIEGUE

Este diagrama permite visualizar los principales componentes utilizados en tiempo de ejecución del sistema. Aparecen dos nodos importantes: Cliente y Azure App Services.

**Nodo Cliente:** Contiene el navegador web encargado de ejecutar los códigos HTML, JavaScript y CSS que permite visualizar la aplicación de forma intuitiva y amigable.

**Nodo Azure App Services:** Este nodo contiene 3 componentes más importantes como son:

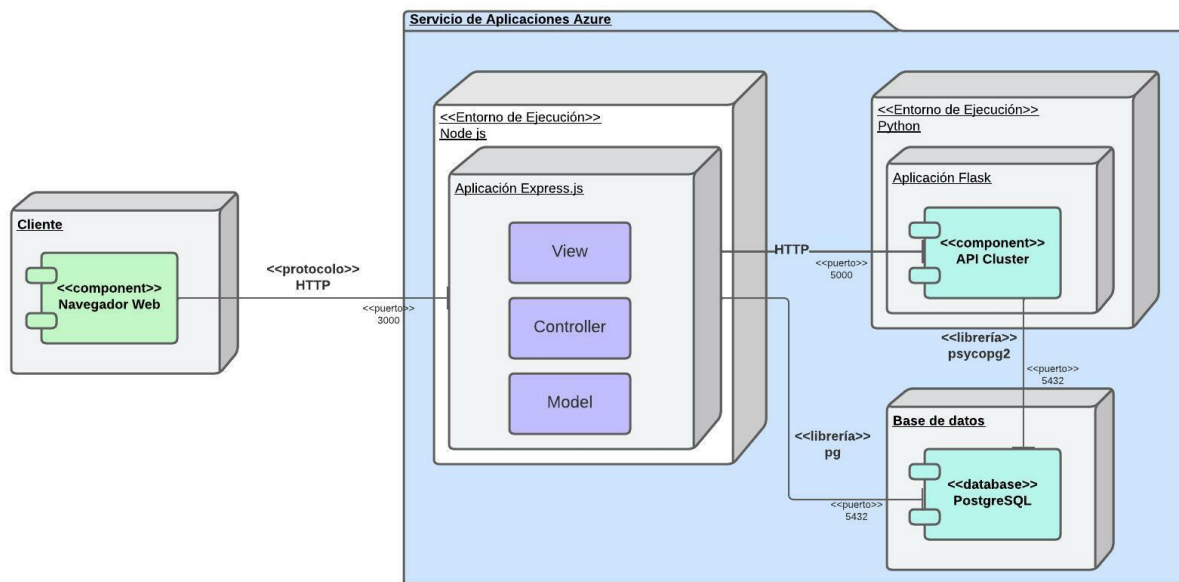
**Node.js:** Es el entorno de ejecución con el framework Express, almacena los 3 componentes más relevantes del sistema: los “View” contiene las vistas y estilos del sistema, “Controller” contiene la lógica del negocio y rutas del sistema, “Model” contiene

la conexión a la base de datos, conexión a la API mediante llamada Http y helpers funciones específicas personalizadas para agregar lógica en las plantillas.

**Python:** Es el entorno de ejecución con el framework Flask, implementados para la API de gestión de los pedidos.

**PostgreSQL:** Es implementado para guardar la información de todo el sistema.

**Figura 77**  
*Diagrama de Despliegue*



*Nota: Diagrama de Despliegue bajo el estándar UML. Elaborado por: Los autores*

## 4.2 DESPLIEGUE DE LA BASE DE DATOS

La Base de Datos, funciona como eje central de la aplicación, para manejar y respaldar cada uno de los datos entrantes y salientes, para su implementación en la red se utilizó el servicio en la nube proveído por Microsoft: **Azure Database for PostgreSQL.**

Este servicio provee un servidor autoadministrado, inteligente y flexible para bases de datos PostgreSQL. El servidor es configurado para la creación de Bases de Datos que almacena la información, además provee disponibilidad para responder a cada de las peticiones que son generados en la aplicación web de InVirtual Store.

El servidor fue creado utilizando la ubicación de los servidores de Azure en la zona East US, misma que comparte el huso horario con Ecuador, con el fin de evitar problemas en acciones que extraigan la fecha y hora actual.

El servidor se configuró con la versión 11 de PostgreSQL, esta versión cuenta con un gran mantenimiento de la comunidad, y una buena gestión en bases de datos grandes, que requieran altas cargas de trabajo.

Respecto al rendimiento, se configuró un servidor de nivel básico, con 1 núcleo virtual y 20 GB para el almacenamiento de datos, además se activó la opción de auto escalamiento, para que el servidor pueda expandirse de acuerdo con lo necesario en casos donde el almacenamiento inicial se vea superado.

Una vez creado servidor, para la creación de la base de datos y carga inicial de datos se procedió a ejecutar las acciones realizadas en la actividad: “**Creación de la BBDD y Carga de Datos**” del Sprint 1, explicadas en el Capítulo 3.

### 4.3 DESPLIEGUE DE LA APLICACIÓN

El despliegue de la aplicación se lo realizó utilizando los servicios de Microsoft Azure. Para poder subir el código, se utilizó la herramienta Azure App Services, que permite compilar, desplegar, implementar y administrar las aplicaciones mediante el uso de contenedores.

El servicio se configuró para ser ejecutado en una máquina virtual Linux que funciona como el servidor virtual de la aplicación, además se estableció el entorno de ejecución del código que va a recibir el servicio con Node.js v16.17.0.

Una vez configurado el servicio, se procedió a la implementación del código de la aplicación en el servicio mediante la utilización de las herramientas proveídas por GitHub Actions, que permite ejecutar acciones de Entrega Continua o CD por sus siglas en inglés.

Mediante las acciones de CD, el repositorio se vincula con el servicio Azure App Services para generar el despliegue automático al servidor a través de GitHub Actions.

Este servicio, usa la sintaxis YAML para definir el flujo de trabajo para el despliegue, este archivo es creado dentro del repositorio de código en la carpeta: `github/workflows`. Cada flujo de trabajo se almacena como un archivo YAML independiente en el repositorio de código. El archivo contiene varias líneas que se explican a continuación:

**name:** Nombre del flujo de trabajo tal como aparecerá en la pestaña 'Actions' del repositorio de GitHub.

**on [acción]:** Especifica el disparador para este flujo de trabajo. Este archivo usa el evento push, por lo que se activa la ejecución de flujo de trabajo cada vez que alguien envía un cambio al repositorio.

**branches:** Especifica el nombre de la rama del repositorio en la cuál debe ser ejecutada la acción establecida en el literal anterior.

**jobs:** Agrupa todos los trabajos que se ejecutan en el flujo de trabajo.

Se ejecuta inicialmente el trabajo (job) de **build** (construcción), que configura el entorno de Node.js, procede a ejecutar la instalación de dependencias y la construye un archivo compilado que puede ser subido posteriormente al servidor.

### Figura 78

*Código de la Construcción de la aplicación en el archivo YAML*

```
build:
  runs-on: ubuntu-latest

  steps:
    - uses: actions/checkout@v2

    - name: Set up Node.js version
      uses: actions/setup-node@v3
      with:
        node-version: '16.17.0'

    - name: npm install, build, and test
      run: |
        npm install
        npm run build --if-present
```

*Nota: Parte de archivo YAML para acciones de CD. Elaborado por: Los autores.*



Una vez culminado el trabajo de construcción se ejecuta el trabajo de despliegue (deploy), estableciendo su dependencia al trabajo previo mediante el comando **needs**, a continuación, se procede a desplegar obteniendo la información del artifact de la aplicación, y conectándolo con las credenciales de Azure App Service para desplegar la aplicación en el servicio creado.

### Figura 79

*Código de la Construcción de la aplicación en el archivo YAML.*

```
deploy:
  runs-on: ubuntu-latest
  needs: build
  environment:
    name: 'Production'
    url: ${ steps.deploy-to-webapp.outputs.webapp-url }

  steps:
    - name: Download artifact from build job
      uses: actions/download-artifact@v2
      with:
        name: node-app

    - name: 'Deploy to Azure Web App'
      id: deploy-to-webapp
      uses: azure/webapps-deploy@v2
      with:
        app-name: 'invirtual'
        slot-name: 'Production'
        publish-profile: ${ secrets.AZUREAPPSERVICE_PUBLISHPROFILE_CBFCE6AF48674C2DB1A2DDB5DE795D7F }
        package: .
```

*Nota: Parte de archivo YAML para acciones de CD. Elaborado por: Los autores.*

Finalmente, al subir el commit con el archivo YAML creado, automáticamente se ejecutó la acción de CD en Github. Una vez culminada la acción, la aplicación está desplegada en el servidor y puede ser accedida en la web a través del siguiente enlace: <https://invirtual.azurewebsites.net/>.

## 4.4 DESPLIEGUE DE LA API DE GESTIÓN DE REPARTICIÓN

Adicionalmente al código de la aplicación principal, fue necesario desplegar el código de la API REST complementaria de la aplicación que permite gestionar los pedidos, mediante técnicas de clustering.

La implementación se la realizó de igual manera utilizando la herramienta Azure App Services. El servicio se configuró de manera similar al explicado en el literal anterior, tan solo cambiando el entorno de ejecución del código con Python 3.8.

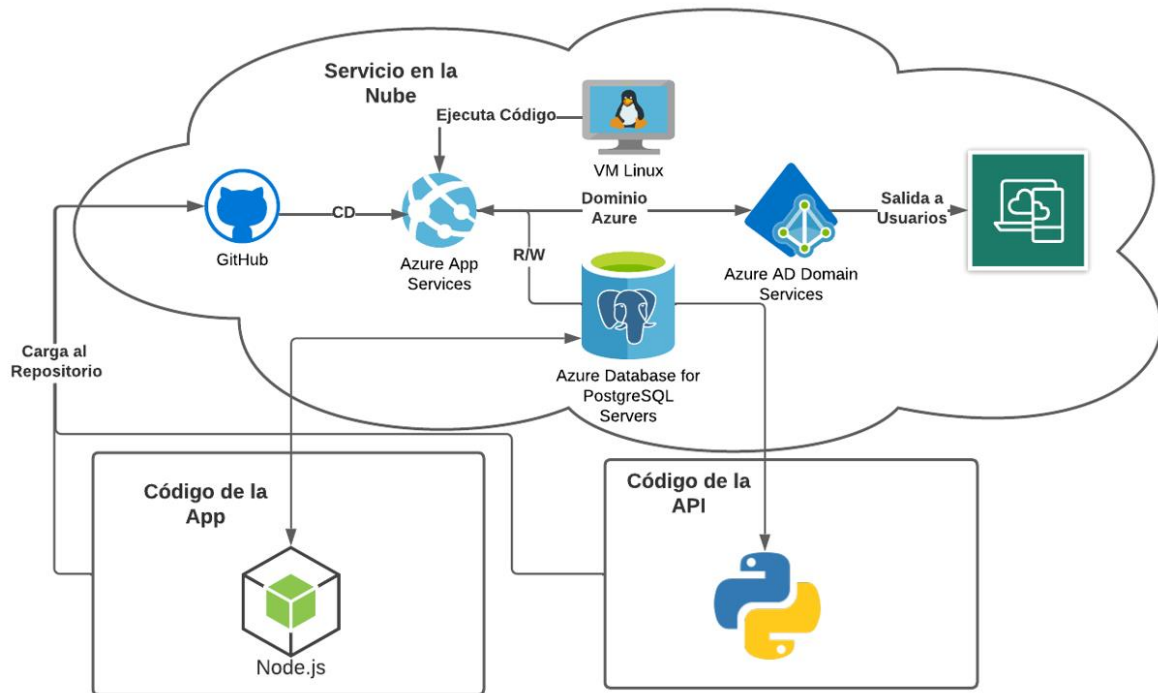
De manera similar a lo realizado con el código principal, se ejecutaron 2 trabajos dentro del archivo YAML creado. El trabajo de build que configura el entorno de Python, creando el entorno virtual y posteriormente ejecutando la instalación de dependencias necesarias para la ejecución del código.

Una vez culminado el trabajo de construcción se ejecuta el trabajo de deploy, donde se realiza el despliegue, obteniendo la información del artifact de la aplicación, y conectándolo con las credenciales del nuevo servicio creado para desplegar la API.

Finalmente, una vez cargado el archivo YAML en el repositorio, se ejecutó la acción de CD en Github. Una vez culminada la acción de CD la aplicación está desplegada en el servidor, y puede ser accedida en la web a través del siguiente enlace: <https://invirtual-api.azurewebsites.net/>.

A continuación, se muestra mediante un diagrama el proceso seguido en las actividades de implementación de cada uno de los servicios necesarios para lograr el despliegue en la nube, y la aplicación completamente funcional está disponible para los usuarios a través de la red.

**Figura 80**  
*Diagrama de Servicios de Implementación*



*Nota: Diagrama que muestra los programas e interconexiones de los softwares y servicios utilizados para la implementación en Azure App Services. Elaborado por: Los autores.*

## 5. CONCLUSIONES

1. Este proyecto permitió desarrollar un software funcional que permite mejorar los procesos de la administración de ventas, y control de inventario en la tienda InVirtual Store. Ahora la tienda cuenta con registros de su inventario y ventas de forma rápida y permanente a través de la web, derivando en un ahorro de tiempo cercano al 50% en gestionar el stock. Adicionalmente, se estima que el porcentaje de pérdidas o confusiones en los registros de los pagos de cada cliente será menor al 1%.
2. Los servicios de Microsoft Azure utilizados en la implementación de la aplicación web, permitió una rápida configuración gracias a su interfaz dinámica e intuitiva. Además, reduce el tiempo destinado para la mejora de la seguridad de la aplicación, ya que, al contar con un sistema auto administrado, y estar soportado por una gran empresa, los servicios utilizados ofrecen varias capas de seguridad que vuelven a la aplicación mucho más fiable, sin la necesidad de realizar revisiones constantes.
3. Las pruebas realizadas en el Capítulo 3, demuestran que el código desarrollado es fiable y seguro, ya que mostró solo advertencias mínimas respecto a la calidad del código, y un porcentaje del 100% de respuestas correctas en cada petición. Estos resultados, se deben al seguimiento de estándares de buenas prácticas en la realización de aplicaciones con Node.js. Estas buenas prácticas guiaron a que se entregue un código fiable a los clientes, y que brinde a los programadores un código comprensible, con facilidad de mantenimiento.

4. La aplicación al ser desarrollada con el paradigma de programación funcional, mas no con el de programación orientada a objetos (OOP), presentó dificultades en la realización de los diagramas de secuencia, necesarios en la etapa de diseño del sistema. Este problema se dio debido a que este tipo de diagrama sigue un estándar UML, mismo estándar que está enfocado principalmente para aplicaciones realizabas bajo el paradigma OOP.
  
5. La utilización de la metodología SCRUM a lo largo del proyecto, impulsó a tener una comunicación clara y precisa con las personas involucradas el sistema, en la culminación de cada Sprint. Esto permitió tener una retroalimentación constante, que derivó en una mayor claridad en el desarrollo requerimientos, logrando así entregar un código con una funcionalidad y diseño enfocado en los aspectos esenciales de las actividades realizadas en InVirtual Store.

## 6. RECOMENDACIONES

1. Para una nueva versión de la aplicación se debe incorporar un dashboard de gráficos estadísticos que abstraigan los datos históricos generados de los pedidos y los pagos realizados en el sistema, permitiendo que mediante la interpretación de los datos que arrojan los gráficos generados, la tienda pueda mejorar en sus operaciones diarias.
2. Para aprovechar y ampliar las funcionalidades que ofrece el sistema, se recomienda conectar la aplicación web con un e-commerce de la tienda, que permita una comunicación bidireccional entre ambos sistemas, para el intercambio constante de información de ventas, stock y clientes entre ambos sistemas.
3. Se recomienda investigar métodos de comprensión de imágenes, o migración de servicios de almacenamiento, con el objetivo de gestionar de mejor manera el alojamiento de las imágenes de los productos registrados en sistema, ya que el número de productos de la tienda irá aumentando de forma gradual, ocasionando que eventualmente el almacenamiento establecido actualmente se vea superado.
4. Se recomienda implementar un framework para la realización de pruebas unitarias de código como Jest o Mockito tanto para el código realizado actualmente, como para futuras actualizaciones de la aplicación. Ya que, contar con este tipo de pruebas permitirá tener una mayor certeza de que las funciones más importantes del sistema funcionan de la manera esperada.

## REFERENCIAS

- Ahmad, S. I., Rana, T., & Maqbool, A. (2022). A Model-Driven Framework for the Development of MVC-Based (Web) Application. *Arabian Journal for Science and Engineering*, 47(2). <https://doi.org/10.1007/s13369-021-06087-4>
- Anne Millen Porter. (1997). *One Focus, One Supply Base Purchasing*.
- Bracey, K. (2018). What is Figma? *Envatotuts*.
- Buddika, K. V. N. (2017, noviembre). *Purchase, Stock and Sales Management System for Gishani Distributors*. <https://dl.ucsc.cmb.ac.lk/jspui/bitstream/123456789/3973/1/0520381.pdf>
- Chase, R. B., & Jacobs, F. R. (2011). Administración de operaciones 13a; Producción y cadena de suministros. En *McGraw-Hill Companies, Inc.* (Vol. 13).
- Cillero, M. (2020). *Diagrama de secuencia*. <https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/diagrama-de-interaccion/diagrama-de-secuencia/>
- Díaz, H., García, R., & Porcell, N. (2012). Las PyMES: costos en la cadena de abastecimiento. *Revista Escuela de Administración de Negocios*, 63(0120–8160).
- Dini, M., Gligo, N., & Patiño, A. (2021). *Transformación digital de las mipymes: elementos para el diseño de políticas*. <https://repositorio.cepal.org/handle/11362/47183>
- Eliyan, L. F., & di Pietro, R. (2021). DoS and DDoS attacks in Software Defined Networks: A survey of existing solutions and research challenges. *Future Generation Computer Systems*, 122. <https://doi.org/10.1016/j.future.2021.03.011>
- Fernando Montalvo-Coronel, L. I., & Hipólito Orozco-Santos, C. I. (2020). Disrupción digital en tiempos de pandemia efectos en el mercado tecnológico en la provincia de Manabí – Ecuador. *Polo Del Conocimiento: Revista Científico - Profesional*, ISSN-e 2550-682X, Vol. 5, Nº. 8, 2020, Págs. 353-375, 5(8), 353–375. <https://doi.org/10.23857/pc.v5i8.1592>

- Giannice, S. G. (2009). Administración y logística en la cadena de suministros. *Revista de Investigaciones Del Departamento de Ciencias Económicas*, 2(3).  
<https://doi.org/10.54789/rince.340>
- Gigleux, A., & Campbell, A. (2016). SonarQube Documentation. *SonarQube Documentation*, Ci.
- Gokhale, S., Turcotte, A., & Tip, F. (2021). Automatic migration from synchronous to asynchronous JavaScript APIs. *Proceedings of the ACM on Programming Languages*, 5(OOPSLA). <https://doi.org/10.1145/3485537>
- González Patricia. (2021, julio 27). *Desarrollos de software para la gestión empresarial | Revista Líderes*. <https://www.revistalideres.ec/lideres/software-gestion-empresas-tecnologia.html>
- K, Dr. I. S., Muyunuddin, K., & N, K. (2022). POINT OF SALES AND INVENTORY MANAGEMENT SYSTEM. *EPRA International Journal of Multidisciplinary Research (IJMR)*, 8(7), 88–91. <https://doi.org/10.36713/epra2013>
- Lara Walter. (2015). *¿Cómo funciona la metodología Scrum? Qué es y sus 5 fases*.  
[https://platzi.com/blog/metodologia-scrum-fases/?fb\\_comment\\_id=871870762906431\\_872603689499805&utm\\_source=google&utm\\_medium=cpc&utm\\_campaign=18798607679&utm\\_adgroup=&utm\\_content=&gclid=Cj0KCQiAnNacBhDvARIsABnDa6\\_vOGZOQ9AWHuFgUNWzg1L5welmPtQurrKxrQDzQ5YNDCIZAbLgDpQaAksEEALw\\_wcB&gclidsrc=aw.ds](https://platzi.com/blog/metodologia-scrum-fases/?fb_comment_id=871870762906431_872603689499805&utm_source=google&utm_medium=cpc&utm_campaign=18798607679&utm_adgroup=&utm_content=&gclid=Cj0KCQiAnNacBhDvARIsABnDa6_vOGZOQ9AWHuFgUNWzg1L5welmPtQurrKxrQDzQ5YNDCIZAbLgDpQaAksEEALw_wcB&gclidsrc=aw.ds)
- Liu, C. (2022). *Research Article Development and Application of Sales System Software Based on Computer Network*. <https://doi.org/10.1155/2022/4524698>
- Lucassen, G., Dalpiaz, F., van der Werf, J. M. E. M., & Brinkkemper, S. (2016). Improving agile requirements: the Quality User Story framework and tool. *Requirements Engineering*, 21(3). <https://doi.org/10.1007/s00766-016-0250-x>



- Luther, D. (2020, septiembre 16). *Retail Inventory Management: What It Is, Steps, Practices and Tips / NetSuite*. <https://www.netsuite.com/portal/resource/articles/inventory-management/retail-inventory-management.shtml>
- Marchant Castelnuovo, A. S. (2012). *Diseño y desarrollo de un sistema WMS (Warehouse Management System) para la empresa Logistecsa bajo la metodología MSF*. <http://repositorio.espe.edu.ec/jspui/handle/21000/5284>
- Martín Romero, E., & esneider.martinr@campusucc.edu.co. (2019). Diseño e implementación de sistema de inventarios para el almacén de pinturas y ferretería Ferrecolor. *Alcívar Dick, F. A. (26 de 04 de 2018). Diseño de Una Herramienta de Productividad: Sistema de Inventario y Facturación Para Microempresas y Pequeñas Empresas. Obtenido de Http://Repositorio.Ug.Edu.Ec/Handle/Redug/29193, 7, 21–25.* <https://doi.org/10.5377/FAREM.V0I7.2629>
- Ochoa, D. L. C. (2021). Influencia del uso de redes sociales en la venta de productos: Microempresa Color Rosa. *REVISTA ERUDITUS*, 2(2), 61–74. <https://doi.org/10.35290/RE.V2N2.2021.459>
- Pedriquez Daleska. (2022, junio 29). *¿Qué es un diagrama de contexto?*
- Plúas Navarrete, F. B., & Ponce Baque, R. A. (2018). *Desarrollo e implementación de un sistema de control y administración de ventas en la empresa global sumec*. <http://dspace.ups.edu.ec/handle/123456789/16998>
- Ramírez Sánchez, G. S., Ortiz Gonzalez, H. J., & García Cruz, K. E. (2022). Comportamiento del consumidor. *TEPEXI Boletín Científico de La Escuela Superior Tepeji Del Río*, 9(17). <https://doi.org/10.29057/estr.v9i17.8083>

- Sandeep, R. C., Sánchez-Gordón, M., Colomo-Palacios, R., & Kristiansen, M. (2022). Effort Estimation in Agile Software Development: A Exploratory Study of Practitioners' Perspective. *Lecture Notes in Business Information Processing, 438 LNBIP*.  
[https://doi.org/10.1007/978-3-030-94238-0\\_8](https://doi.org/10.1007/978-3-030-94238-0_8)
- Singh, M. S. (2020). MVC Framework: A Modern Web Application Development Approach and Working. *International Research Journal of Engineering and Technology*.
- Sordo Ana Isabel. (2021, febrero 25). *Metodología Scrum: qué es, cuáles son sus fases y cómo implementarla*. <https://blog.hubspot.es/marketing/metodologia-scrum>
- Statista Resource Center. (2022). *Enterprise Software - Worldwide | Statista Market Forecast*.  
<https://www.statista.com/outlook/tmo/software/enterprise-software/worldwide>
- Sudarsono, B. G. (2020). Adopting SCRUM Framework in a Software Development of Payroll Information System. *International Journal of Advanced Trends in Computer Science and Engineering, 9(3)*. <https://doi.org/10.30534/ijatcse/2020/17932020>
- Trogh, N., Downer, R., Nallamothe, N., & Spence - Burnard, M. (2022, diciembre 20). *What is Azure Load Testing Preview?* <https://learn.microsoft.com/en-us/azure/load-testing/overview-what-is-azure-load-testing>