UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Sistema Web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.

Trabajo de graduación como modalidad de Trabajo Profesional presentado por Santos Jeremías García Tzul para optar al grado de Licenciado en Tecnología de Sistemas Informáticos

Guatemala,

2021

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

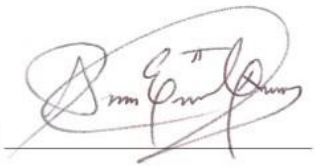
Sistema Web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.

Trabajo de graduación como modalidad de Trabajo Profesional presentado por Santos Jeremías García Tzul para optar al grado de Licenciado en Tecnología de Sistemas Informáticos

Guatemala,

2021

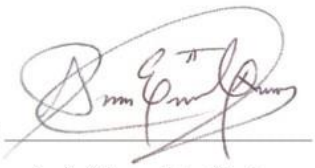
Vo.Bo.:



(f) _____

Ing. Paúl Eleazar Cotom Xicará

Tribunal Examinador:



(f) _____

Ing. Paúl Eleazar Cotom Xicará

(f) _____

Ing. Edwin Daniel Arturo Recinos Armas

(f) _____

MBA. Eddy Omar Arreaga López

Fecha de aprobación: Guatemala, 21 de junio de 2021.

CONTENIDO

LISTA DE CUADROS

LISTA DE FIGURAS

RESUMEN

En la actualidad la tienda cuenta con la deficiencia al mostrar los artículos que disponen ya que estos están almacenados, dificultando la búsqueda según las especificaciones deseadas y de tal manera que a un cliente se le muestra un promedio de cinco artículos donde debe tallar una por una hasta encontrar el que desea. Este sistema ayudará a mejorar la adquisición de los artículos dentro de la tienda de ropa típica a través de un sistema web que cuenta con un catálogo de los artículos y un algoritmo de recomendación a los artículos complementarios según las especificaciones requeridas del cliente para completar el traje, empleando el lenguaje de programación PHP y almacenando los datos a través del gestor MySQL de manera local. Esta será implementada a través de un dispositivo dentro de la tienda para que los clientes puedan visualizar de mejor manera cada artículo y hacer uso del algoritmo de recomendación de manera automática.

Dicho sistema contempla los procesos de ingeniería de software, en su modelo de prototipos que son: recolección y refinamiento de requisitos, modelado diseño rápido, construcción del prototipo, desarrollo, evaluación del prototipo por el cliente, refinamiento del prototipo y la estructura de los datos serán basados en el modelo relacional para facilitar la búsqueda de estos. Será una herramienta dinámica ya que se podrán manipular las imágenes dentro del catálogo.

I. INTRODUCCIÓN

El proceso de atención de la tienda de ropas típicas, dentro de la investigación realizada se determinó una deficiencia en el servicio, ya que los artículos que ofrecían a sus clientes no estaban identificados y no disponía de alguna herramienta que pudiera ayudar a la gestión y visualización de los artículos en existencias. Por tal razón se desarrolló un prototipo de un sistema de catálogo personalizado para ayudar a los clientes en la búsqueda de los artículos que necesita y en la gestión de los productos de la tienda. Dentro de los procesos que se realizan en el desarrollo del sistema, se detalla la metodología empleada en el progreso del prototipo, desarrollados en el lenguaje de programación PHP del lado del servidor y con JavaScript del lado del cliente que ayudan a completar los objetivos planteados.

Se detalla cada uno de los procedimientos a realizar desde le recopilación de información, modelado, construcción, desarrollo, refinamiento del prototipo, presentando los requerimientos del sistema, su funcionamiento y lo no funcional el diseño de la interfaz y en el diseño de la lógica de programación que se emplea para cada una de las funcionalidades desarrolladas.

II. OBJETIVOS

A. GENERAL

Codificar un sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas para facilitar la adquisición de estos con las herramientas de desarrollo PHP y MySQL.

B. ESPECÍFICOS

1. Implementar el análisis y diseño del sistema web según los requerimientos para la codificación del proceso para el sistema web.
2. Implementar el modelo de prototipos como modelo de desarrollo para el sistema web que ayude administrar los procesos en éste.
3. Desarrollar el interfaz del usuario con el Framework Bootstrap para la facilidad de su uso.

III. JUSTIFICACIÓN

Al atender a un cliente en la tienda este pregunta por el color, tipo de forma, material del cual está elaborada, el problema radica en la falta de visualización de los artículos y ésta dificulta dar con el producto específico deseado por el cliente, ya que en la actualidad muestran un promedio de cinco artículos donde se talla una por una hasta encontrar el correcto y ésta consume mucho tiempo al atender a un cliente aún más si necesita combinar con otro artículo, ya que deberán realizar el mismo procedimiento detallado con anterioridad, como bien se sabe que al momento de adquirir un traje típico es un poco más complejo seleccionar los productos como lo es un corte, blusa, faja y zapato para completar un traje. La solución al mismo es ayudar al cliente a poder visualizar cada artículo a través de imágenes dentro del sistema y poder combinar el traje de mejor manera con la ayuda de un algoritmo de recomendación según las especificaciones del cliente.

Viendo la necesidad de la tienda de poder ayudar a sus clientes en la adquisición de su traje y en la administración adecuada para sus productos se desarrolla el sistema de catálogo personalizado, con la ayuda del algoritmo de referencia reducirán el tiempo en la atención y la cantidad de productos a mostrar por cliente. El sistema web estará conformado por varios módulos como son: un login para permitir a los dueños de la tienda a registrar la mercadería, un formulario y subir una fotografía de cada producto almacenado dentro de la base correspondiente y como último módulo la del catálogo, donde estará dividida por categorías, al momento de visualizar un producto el sistema pueda recomendar otros artículos relacionados según las especificaciones de forma automática. Dicho sistema será implementado dentro de la tienda de manera local.

IV. MARCO TEÓRICO

A. CONCEPTOS DE SISTEMAS DE CÓMPUTO

1. Sistema operativo.

El sistema operativo es el software básico que controla una computadora (Moreno Pérez, 2012:60), es la base principal que controla todos los componentes electrónicos, es quien gestiona las instrucciones según las acciones del usuario a través de los periféricos denominados hardware por su nombre en inglés; el sistema es el mediador de los software que son instaladas y que cada instrucción será gestionada por el sistema operativo.

Las funciones del sistema operativo son las siguientes:

1. Proporciona una interfaz amigable conocida como IU, donde nos facilita el acceso a los diferentes recursos que se encuentra almacenados dentro de las unidades de almacenamiento.
2. Gestiona el hardware de la computadora y asigna las funcione necesarias para completar con la tarea que el usuario pretende elaborar.
3. Administra el flujo de los procesos y subprocesos de los programas que están en ejecución.

a. Tipos de sistemas operativos

Los sistemas operativos están clasificados por las características y funcionalidades, se desglosan en sistemas operativos monousuario, multiusuario, monotarea y multitarea.

1) Sistema operativo monousuario

Estos tipos de sistemas operativos cuentan con la característica de que un único usuario puede utilizar tras su ejecución, ejemplo de estos son los sistemas operativos Windows (XP, Vista,7, 8 y 10).

2) Sistemas operativos multiusuarios

Son los tipos de sistemas operativos que pueden acceder más de un usuario a la vez para utilizar los recursos de hardware, ejemplo de los sistemas tenemos Windows Server y Linux.

3) Sistema operativo monotarea

Fueron basados mediante el único procesador que se tenía y ejecutaba un único proceso, el ejemplo de este sería DOS.

4) Sistemas operativos multitareas

Son todos los sistemas operativos que pueden ejecutar N procesos dentro del sistema, lo sistemas operativos que cumplen con estas características son, Windows, Linux y Mac Os.

b. Sistemas operativos actuales

En la actualidad se emplean tres principales que son: Windows, Unix/Linux y Mac OS; estos son los utilizados por la comunidad. Cada uno trabaja de distintas maneras en la gestión de memoria, administración de los procesos y estos sistemas están diseñados para determinadas funciones como los veremos a continuación.

1) Sistema operativo Windows

Microsoft decidió diseñar un nuevo sistema operativo basándose en la experiencia y buenos resultados de su entorno gráfico (Morera Pascual & Pérez-Campanero Atanasio, 2002:52), Windows se convirtió en uno

de los sistemas operativos estándar para todas las computadoras comerciales, por su fácil manejo, estabilidad, portabilidad, compatibilidad, rendimiento e interfaz de usuario empleando el diseño en ventanas. Este sistema operativo cuenta con seis versiones más reconocidas siendo estas populares.

Windows 95: fue una versión del sistema operativo donde se incluyó nuevas funcionalidades como el botón de inicio, la barra de tareas entre otros y fue presentada en el año 1995.

Windows XP: uno de los mejores sistemas operativos diseñados en su época, contaba con un interfaz de usuario en ventanas, implementando nuevas funciones que ayudaron a magnificar las conectividades a través de las redes inalámbricas, asistencia remota y la más reconocida de esta Windows Movie Maker, presentada en el año 2001.

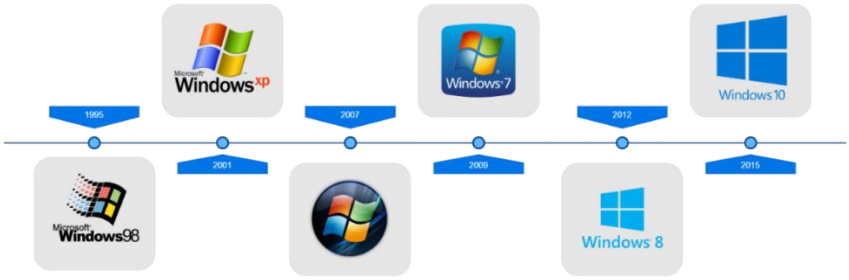
Windows vista: fue una actualización al interfaz de usuario de la versiona anterior Windows XP, además de esta contaba con muchas deficiencias de seguridad, la administración del recurso de las computadoras; muchos de los usuarios tacharon esta versión en la lista de los sistemas operativos con más problemas, presentada en el año 2007.

Windows 7: sin duda uno de los sistemas operativos más reconocidos por los usuarios e internautas, mejorando el interfaz de usuario a través del efecto Aero Glass, el rendimiento óptico para la mayoría de las computadoras en su momento y mejorando los problemas de seguridad en la versión anterior Windows vista, presentada en el año 2009.

Windows 8: dentro de esta versión surgieron varias modificaciones, entre estas que podemos mencionar el diseño del interfaz de usuarios con nuevos efectos planos en las ventanas, el menú inicio fue el mayor cambio al ser eliminada de la barra de tareas y adaptada para los equipos táctiles, presentado en el año 2012.

Windows 10: es la versión más utilizada actualmente para los equipos, cuenta con un mejor rendimiento, siguiendo la misma línea en el interfaz de usuario con los efectos planos, la integración de Cortana, la sustitución de internet Explorer por Microsoft Edge entre otras funcionalidades que hace que esta versión del sistema operativo sea la estandarizada en los equipos, presentado en el año 2015.

Ilustración 1. Línea de tiempo de sistemas operativos de Windows



2) Sistema operativo Unix/Linux

Es un sistema operativo libre de uso personal o empresarial ya que cuenta con diversas distribuciones como Debian, Ubuntu, Fedora, Arch Linux, Tailis entre otros para su uso personal y distribuciones para empresas que se implementan como sistemas operativos para servidores están: Ubuntu, CentOs, Debian, Arch Linux, Slackware Linux son las que se emplean para administrar los recursos dentro de estas.

Su principal ventaja son las distribuciones que están desarrolladas para diferentes ambientes de trabajo, es un sistema operativo modular que puede ser modificado por la comunidad y poder adecuar a otras áreas específicas.

Ilustración 2. Distribuciones de sistema operativo Unix/Linux

3) Sistema operativo Mac OS

Mac Os (Macintosh Operating System, Sistema Operativo de Macintosh) es el nombre del sistema operativo creado por Apple para sus ordenadores Macintosh (Niño Camazón, 2011:48), es un sistema operativo especial para los equipos de la empresa Apple, su diseño de interfaz es sencillo para la manipulación empleados principalmente por las industrias de la música, desarrolladores de software, estudios de cinematografía y otras.

Ilustración 3. Icono de sistema operativo Mac Os

2. Software.

El software es un conjunto de instrucciones desarrollados para dar solución a una necesidad del usuario (Moreno Pérez, 2012:64), diseñado como una herramienta que permite al usuario realizar una tarea específica de manera computarizada, estos son utilizados en el diario vivir, como lo es en las oficinas con el paquete ofimática y todos aquellos que son implementados dentro de las empresa o entidades.

a. Tipo de software

El software se deriva en dos ramas muy importantes donde ejerce una función específica acorde al problema que da solución, haciendo el uso de manera personal o a un conjunto de usuarios.

1) Software estándar

Son todas aquellas que están pensados en dar solución a un problema específico de manera general, como lo son la suite de ofimática de Microsoft que incluyen (Word, PowerPoint, Publisher, Excel), Zoom, editores de videos y entre otros; que su función es brindar solución a una necesidad compartida entre los usuarios.

Ilustración 4. Ejemplos de software en el mercado

2) Software a medida

Son todas aquellas desarrolladas para realizar una tarea específica que solo cubra las necesidades de una empresa, entidad o un estado; estos siendo específicos y únicos el coste para su desarrollo es elevado.

b. Licencias de software

Las licencias de software son contratos que es establecida por el creador de éste para que los usuarios puedan utilizar según las condiciones establecidas, se otorga un permiso por un periodo de tiempo, cantidad de usuarios, distribución y la utilización adecuada del mismo, nos centraremos en los dos tipos de licencias.

1) Licencias pagadas

Engloba el software que limitan a ciertas funciones o pruebas en determinado tiempo, estos requieren de una suscripción de manera mensual o anual para poder gozar de todas las funcionalidades, ejemplos de estos la suite de Microsoft, antivirus como lo es ESET, Zoom, IBM Watson.

Algunas de las licencias de paga son los siguientes:

- **Freeware:** es software gratuitos, pero limita su uso de manera empresarial.
- **Shareware:** es un software gratuitos y estos pueden contener limitaciones o una prueba por tiempo.
- **Software comercial:** estos software son desarrolladas de manera que se lucren de esta.
- **Adware:** son todos aquellos que vienen incluidos con publicidad que para deshabilitar será necesario comprar el software.
- **Trial:** son los tipos de software que presta todas las funciones por un determinado tiempo, que de ser esencial será necesario comprar el software y reinstalar de nuevo.
- **Software propietario:** son aquellas desarrollados de manera personal, de ser necesario su adquisición deberán ser solicitado el permiso al propietario.

2) Licencias libres

Estos software están diseñados para dar solución a un determinado problema que en su mayoría reciben donaciones por grandes empresas, podemos mencionar a los editores de texto, Gimp, OpenOffice, Mozilla Firefox entre otros que existen en el mercado.

Esto son algunas de las licencias libres comunes:

- **Software libre:** son los software que se pueden distribuir, utiliza y copiar. Algunas de estas vienen instaladas en algunos sistemas operativos.
- **GPL:** es una licencia pública general GNU, estos van vinculados con los programas que incluyan el núcleo del sistema operativo Linux.

- **Debian:** para esta licencia es necesario tener en cuenta lo siguientes. La redistribución, el código fuente de este debe ser incluido y distribuidos, la licencia no puede infectar a un software, deben tener resecciones en la redistribución del código fuente, no puede discriminar a ninguna persona o grupo, los lineamientos presentados no dependerán de donde se llegan a descargar.
- **Open Source:** es una derivada de la licencia Debian.

3. Servidor.

Es una computadora que brinda su hardware para almacenar información que se comparte dentro de una red interna o en la web, como define (Gómez López, 2014) "El servidor es el centro del sistema", ya que son implementadas para almacenar los datos de todos los sistemas que se alojan dentro de ésta, como lo son los registros de usuarios, permisos, ficheros y todo aquello que administra un sistema. Un servidor requiere de un sistema operativo en específico para administrar las peticiones que realicen los usuarios, las configuraciones entre otros, entre estos están Windows Server y distribuciones de Linux.

Tipos de servidores utilizadas actualmente:

- **Servidor web:** es quien presta sus recursos para almacenar distintas páginas web con sus configuraciones correspondientes.
- **Servidor dedicado:** es el que presta sus recursos de almacenamiento para un solo cliente, denominada servidores exclusivos.
- **Servidor compartido:** es lo contrario al dedicado ya que este recibe peticiones de diversos usuarios.
- **Servidor de correo electrónicos:** es el encargado de gestionar todas las peticiones del usuario mediante el envío y recepción de los correos.
- **Servidor de base de datos:** es el servidor que se encarga de gestionar la información enviada desde los sistemas de una empresa o entidad para ser almacenada.
- **Servidor de archivos:** es un servidor encargado de almacenar archivos que serán compartidos dentro de una red.
- **Servidor DNS:** es el encargado de trasladar una dirección IP a un nombre de dominio, teniendo una IP 192.168.0.1 la convierte en (dominio.com) al ser consultada por el usuario.

4. La red de computadoras.

Una red en el área de informática comprende la conectividad entre computadoras como lo expresa (Rivera Darín, 2016:6). Una red se compone de dos o más computadoras conectados entre sí a través de cables de modo que puedan compartir recursos.

Su objetivo principal es compartir recurso entre los equipos conectados, en la actualidad ya podemos agregar los dispositivos móviles, tabletas e inclusive televisores inteligentes. La necesidad de crear una red de computadoras está en compartir los mismos datos a distintos usuarios, presentar un mismo sistema para que estos puedan acceder de manera rápida y gestionar ciertos recursos de los equipos conectados entre sí.

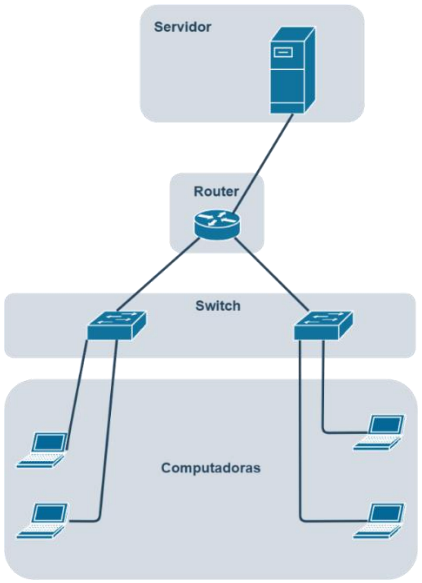
La conexión se establece mediante un direccionamiento denominada IP que simplemente es un valor identificativo como un número de teléfono único dentro de la red de computadoras.

Los dispositivos de red ayudan a controlar, enrutar y almacenar la información que se envié dentro de la red. Estos dispositivos pueden ser:

- **Switch de red: c**onocido también como un conmutador, es un dispositivo de interconexión entre los equipos dentro de la red, su función es poder distribuir a través de sus puertos la conectividad que se tiene formando así una red local denominada LAN.

- **Router de red:** conocida como enrutador, es el dispositivo que enruta los paquetes a su lugar de destino y registrando cada una de estas en una tabla de enrutamiento. Su función es establecer el camino más corto al destino al cual necesite llegar el paquete, lo podemos comparar con un aeropuerto donde nos asigna un numero de vuelo según sea nuestro destino, claro que es una forma de entender el trabajo que debe realizar este dispositivo.
- **Cables:** para tener comunicación entre estos dispositivos es necesario tener una conectividad de manera física para poder comunicarse, existen dos tipos de cables.
    - **Cable de fibra óptica: e**s un cable elaborado de fibras flexibles que permite la transmisión a una distancia de diez kilómetros como máximo mediante la luz.
    - **Cable UTP:** conocidas como cables de red, que están conformado por par de cables trenzados de cobre que ayudan al envió de datos a través de impulsos eléctricos y su capacidad de transmisión es de 100 metros.
- **Access point:** es un punto de acceso para la red inalámbrica que permite la intercomunicación entre estos dispositivos.
- **Un servidor:** es una computadora compartida dentro de la red para almacenar información, gestionar sistemas para uso específicos.

Ilustración 5. Ejemplo de los dispositivos dentro de la red



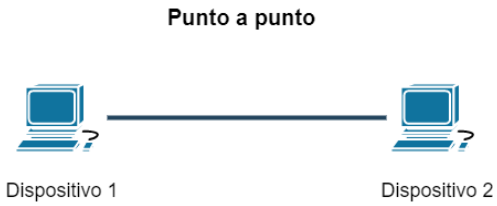
a. Topologías de red.

La forma de interconectar las estaciones de red local, mediante un recurso de comunicación, es decir la estructura topológica de la red, es un parámetro primario que condiciona fuertemente las prestaciones que de la red puede obtenerse (Riera García & Alabau Muñoz, 1992:291). En el ámbito de las redes se maneja la estructura de la topología de red, siendo la base de cómo se interconectan y de qué manera los datos actúan al momento de ser enviados dentro de ésta, a continuación, describimos algunas topologías empleadas en la actualidad.

1) Topología punto a punto

Se trata de una sola conexión entre un dispositivo y otro, es uno de los principios fundamentales para las siguientes topologías.

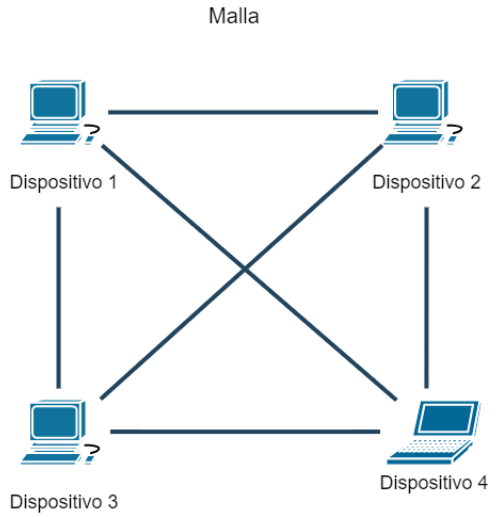
Ilustración 6. Diagrama de topología punto a punto



2) Topología malla

Dentro de esta topología todos los dispositivos tienen que estar conectada entre sí para poder comunicarse, su principal desventaja por cual en la actualidad no se implementa dentro de las redes estándares es su costo elevado, la escalabilidad dificulta su configuración y mantenimiento.

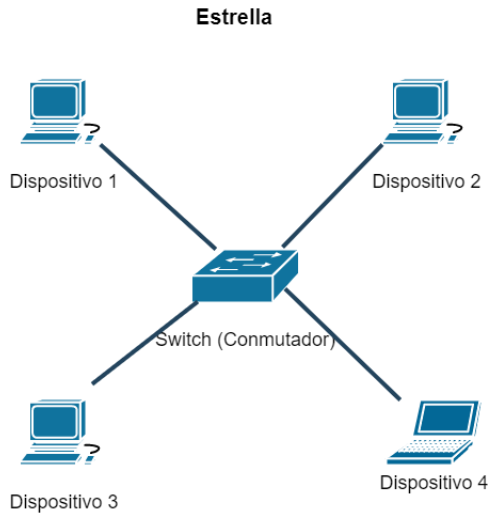
Ilustración 7. Diagrama de topología malla



3) Topología estrella

Denominada como topología de nodos ya que cada dispositivo que necesite interconectarse con otros debe estar conectada de manera física y el nodo debe actuar como un conmutador para que los dispositivos se puedan comunicar entre estos.

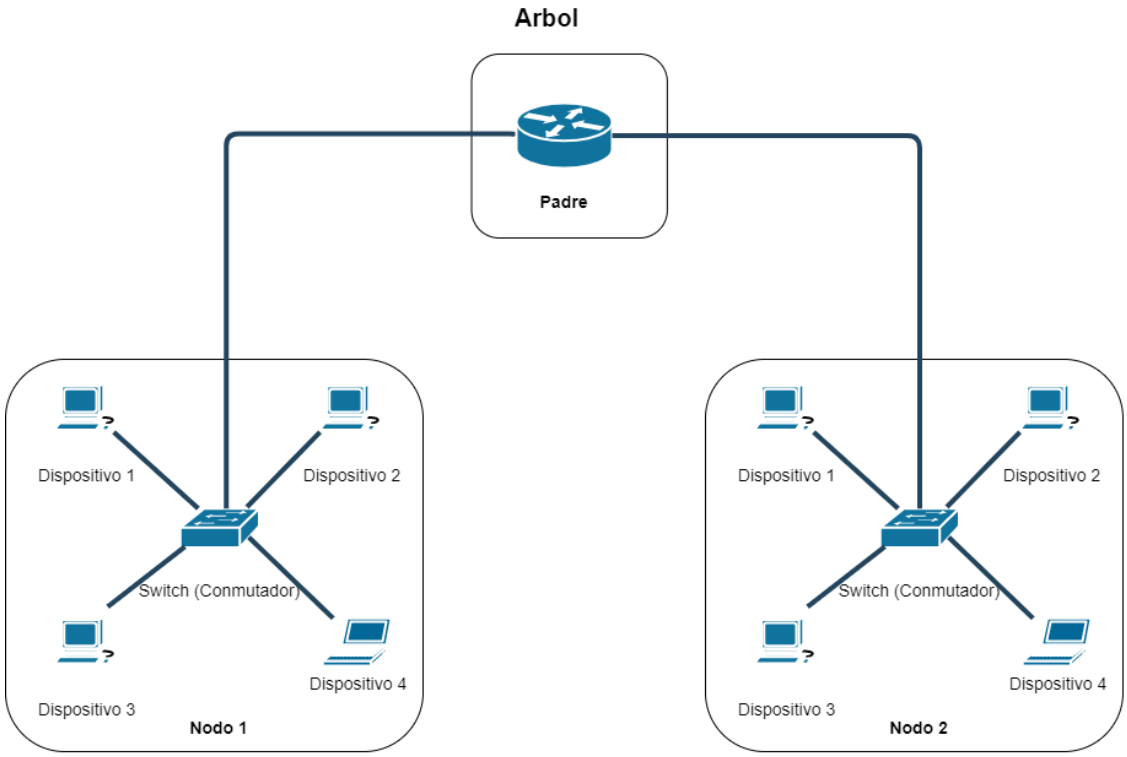
Ilustración 8. Diagrama de topología estrella.



4) Topología de árbol

Conocida como la topología jerárquica donde están compuesto por diversos nodos denominados hojas (Padre o hijo), siendo esto una estructura por nodos, de haber fallos dentro de un nodo específico no afectara a los demás que están conectadas al nodo padre. Para su comunicación es necesario la utilización de los dispositivos de red como conmutador, enrutador y las mencionadas en el apartado dispositivos de la red para poder intercomunicar los nodos.

Ilustración 9. Diagrama de topología de árbol.



5. Base de datos.

Una base de datos es un conjunto de datos persistente que es utilizado por los sistemas de aplicación de alguna empresa dada (J. Date, 2001:10), también podemos interpretar que son datos recolectados mediante software específicos y almacenados para ser consultados, modificados e inclusive eliminadas. Este proceso de la recolección de los datos se realiza en tiempo real ya que puede haber modificaciones dentro de los registros previamente almacenados. Así como se puede observar en los sistemas hospitalarios ya que es necesario registrar cada actividad del paciente en su ficha clínica, de esta manera se lleva un historial de medicamentos, enfermedades entre otros; ya que con estos datos almacenados se pueden estudiar múltiples factores que pueden ayudar a comprender ciertos comportamientos a través de cálculos matemáticos.

a. Modelo conceptual entidad relación

Es un modelo que se aplica a los datos que serán almacenados dentro de la base de datos, en este modelo las entidades tienen una relación con otras entidades, teniendo como objetivo evitar la redundancia de datos.

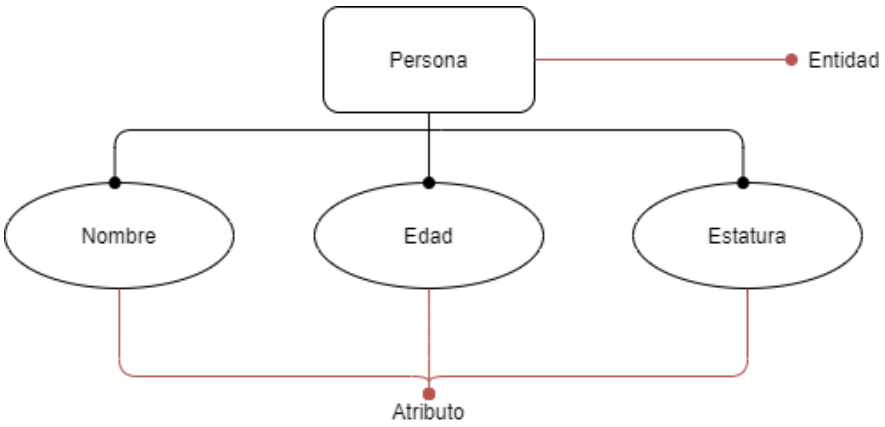
1) Entidad

Una entidad denominada objeto, básicamente es un conjunto de datos que son partes de un solo área, el objeto persona cuenta con múltiples datos como lo son: el nombre, edad, estatura y apellido, son algunas de las que podemos mencionar.

Atributo: es denominada a la característica que tiene el objeto, como se menciona en el ejemplo anterior sobre la entidad persona cuenta con los atributos, nombre, edad, estatura.

Valor: son los datos que se asigna al atributo, estos datos son los recolectados por los sistemas, siguiendo con el ejemplo los valores para los atributos de la entidad serían: juan, 15, 150, García.

Ilustración 10. Estructura de una entidad

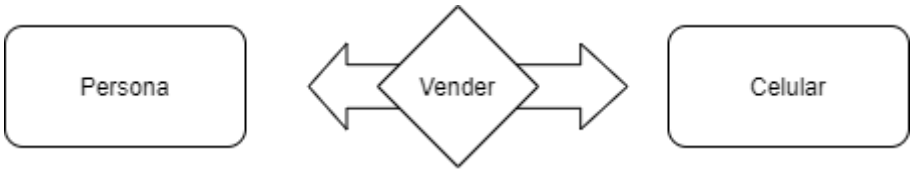


b. Relación y cardinalidad

1) Relación

Para este modelo es muy importante definir las relaciones, ya que nos ayuda a definir dependencia entre las entidades y de esta manera comparten ciertos atributos, de manera más clara podemos decir que las relaciones hacen referencia a un valor específico de una entidad con la otra a fin de poder tener un vínculo.

Ilustración 11. Diagrama de relación de dos entidades



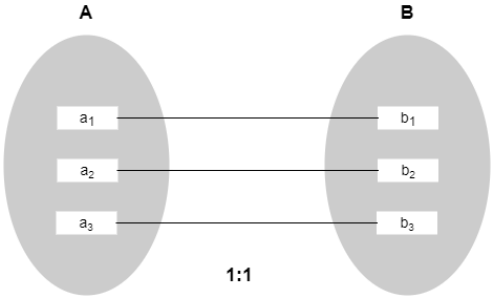
2) Cardinalidad

Expresa la cantidad de relación que cuentan con cada una entidad A con una entidad B, esto nos ayuda a determinar la cantidad de relación que tendrá cada una.

a) Uno a uno

Una entidad de A solo puede tener una relación con una de la entidad B y una entidad de B solo puede tener relación con una entidad de A.

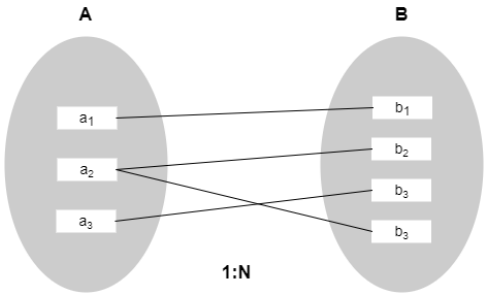
Ilustración 12. Ejemplo de cardinalidad uno a uno



b) Una a muchas

Una entidad de A puede tener relación con varias entidades de B y una entidad de B solo puede tener una relación con una entidad de A.

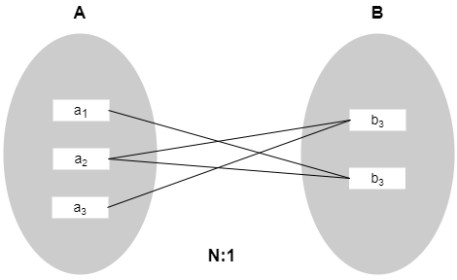
Ilustración 13. Ejemplo de cardinalidad una a muchas



c) Muchas a una

Una entidad de A tiene relación con una sola de la entidad B y la entidad de B puede tener varias relaciones con la entidad de A.

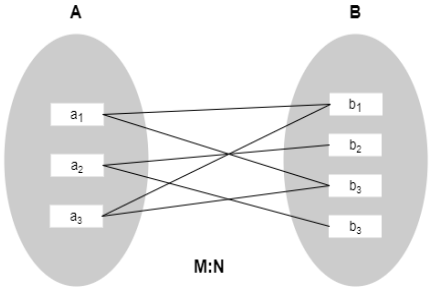
Ilustración 14. Ejemplo de cardinalidad muchas a una



d) Muchas a muchas

La entidad A puede tener varias relaciones con B y una entidad de B puede tener varias relaciones con la entidad de A

Ilustración 15. Ejemplo de cardinalidad mucha a muchas



3) Dependencia de existencia

La dependencia de la existencia de una entidad respecto a otro es importante resaltar para evitar errores, si una entidad A depende de una B entonces al crear o eliminar B se reflejará con A estas acciones tomadas.

Para que las relaciones puedan ser empleadas de manera comprensible es necesario asignar llaves como identificación de que es un valor dependiente o subordinado.

4) La llave primaria

Es un atributo seleccionado de una entidad que diferencia respecto a las demás entidades existentes, una llave primaria es conocida comúnmente como (PK) Primary Key por sus siglas en ingles.

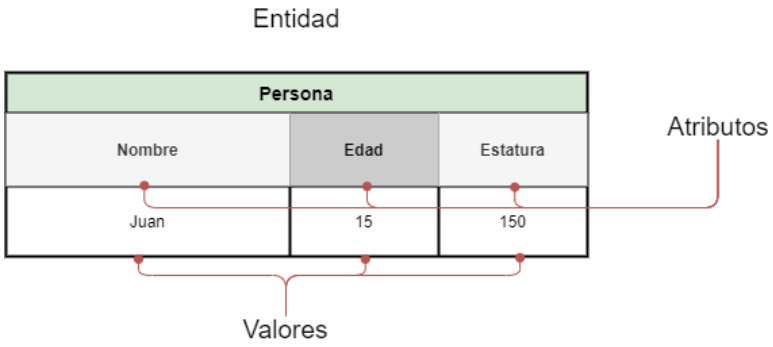
5) La llave foránea

Es una llave que identifica la relación entre las entidades, de la entidad A tenemos una llave primaria “carnet” y esta tiene relación con una entidad B, donde se define está clave foránea que identifica a la entidad A “A_carnet”.

6) Diseño de tablas

Una base de datos que emplea el modelo debe convertir a cada una de las entidades a tablas con el nombre identificativo que le corresponde a la entidad, los atributos son orientadas en las columnas y dentro de las filas serán almacenadas los valores de cada atributo.

Ilustración 16. Estructura de la tabla



7) Normalización de los datos

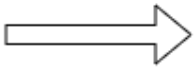
Es uno de los fundamentos para el diseño de una base de datos ya que es necesario estandarizar ciertos atributos, con la finalidad de evitar la redundancia de los datos, incoherencia y la ineficacia. Con este proceso pretendemos estructurar los datos para su uso adecuado según las necesidades de la empresa, mantenimientos y sobre todo pensar en la escalabilidad del software.

8) La unicidad de datos en el atributo

Cada atributo de una tabla debe tener únicamente un tipo específico de datos que almacena. La forma de normalizar los atributos de acorde a lo necesitado lo veremos en este ejemplo donde una empresa brinda servicios para Latinoamérica; donde necesitan normalizar la tabla para almacenar los números de celular.

Ilustración 17. Ejemplo de unicidad de los atributos

Usuario		
nombre	pais	cel
Juan	gt	+50201010101



Usuario			
nombre	pais	area_cel	n_cel
Juan	gt	+502	01010101

Podemos observar que en la primera tabla no está estandarizada el número de área y el número de celular estos van dentro del mismo atributo, esto nos puede generar ciertos conflictos cuando necesitemos realizar alguna consulta específica sobre los datos. En el caso de la tabla del lado derecho tenemos aplicada la estandarización que nos ayudará a identificar de mejor manera el número de área respecto al número de celular.

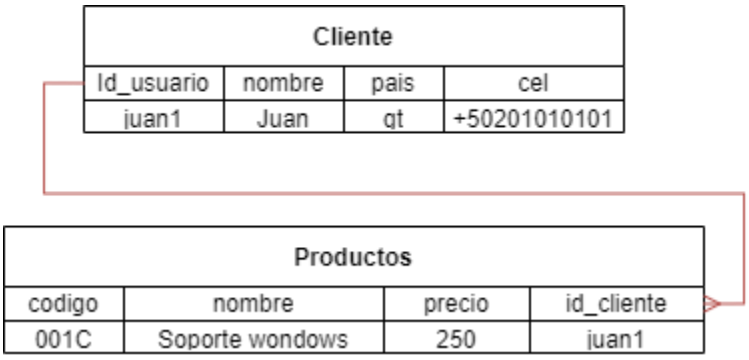
9) Clave principal

Cada tabla debe contener una llave primaria donde identifica el atributo que tendrá la relación con otras entidades tal y como se definió anteriormente.

10) Dependencia funcional

Se basa mediante las relaciones que existen entre las entidades, para las dependencias debemos establecer la llave de una entidad A que relaciona con la entidad B una descripción concreta de la entidad A.

Ilustración 18. Diagrama de dependencia



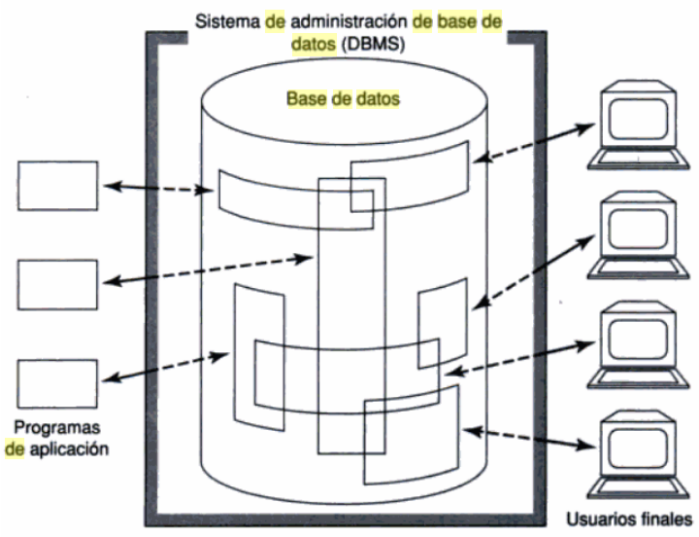
11) Independencia de los atributos

En este apartado debemos tener en cuenta que las llaves primarias y foráneas nunca deberán ser modificados al momento de realizar cualquier actualización en los registros realizados, esto con la finalidad de evitar la incoherencia de los datos.

6. Sistema de base de datos.

Un sistema de base de datos es una herramienta donde podemos almacenar datos, así como le define (J. Date, 2001:5) un sistema de base de datos es básicamente un sistema computarizado para guardar registros. Como bien se menciona se trata de un sistema computarizado que es instalado dentro de un servidor para luego recuperar datos, registrar, editar o eliminar.

Ilustración 19. Estructura de un DBMS



a. Gestor de base de datos MySQL 10.4.17v

MySQL es un gestor de base de datos de tipo relacional con una licencia Open Sourse y como también una licencia GPL, es el encargado de almacenar los datos según sean requeridas por los usuarios, es un sistema empleado para los cliente-servidor.

Las ventajas que nos ofrece MySQL al momento de utilizar en un proyecto web son las siguientes:

- Es un gestor de base de datos gratuita.
- El rendimiento que ofrece es versátil.
- Cuenta con seguridad ya que encripta la contraseñas y como los privilegios para los usuarios de la base de datos.
- Utiliza pocos recursos del hardware donde se encuentra instalada.

1) Estructura de MySQL

Dentro del funcionamiento del gestor de base de datos podemos ver los siguientes procesos que realiza para gestionar los datos.

- **Arquitectura:** definen cada uno de los subprocesos que se deben ejecutar al momento de emplear alguna instrucción.
- **Conectores:** son aquellos sistemas o software que se interconecta entre sí, mediante una conexión.
- **Utilidades:** herramientas que nos ayudan a la administración adecuada.
- **Gestor de recursos:** es el encargado de asignar los recursos necesarios para las conexiones establecidas con el servidor.
- **Interfaz MySQL:** donde se encuentra el intérprete de los comandos para la manipulación y control de las peticiones del usuario.
- **Procesar:** donde se gestionan las peticiones de los clientes como son las inserciones, consultas u otras.
- **Optimizador:** emplean los procesos de manera que los recursos necesarios para estos sean mínimos.
- **Cachés:** área de memoria asignada para almacenar las consultas realizadas.
- **Sistema de ficheros:** área donde se almacenan los distintos objetos de la base de datos.

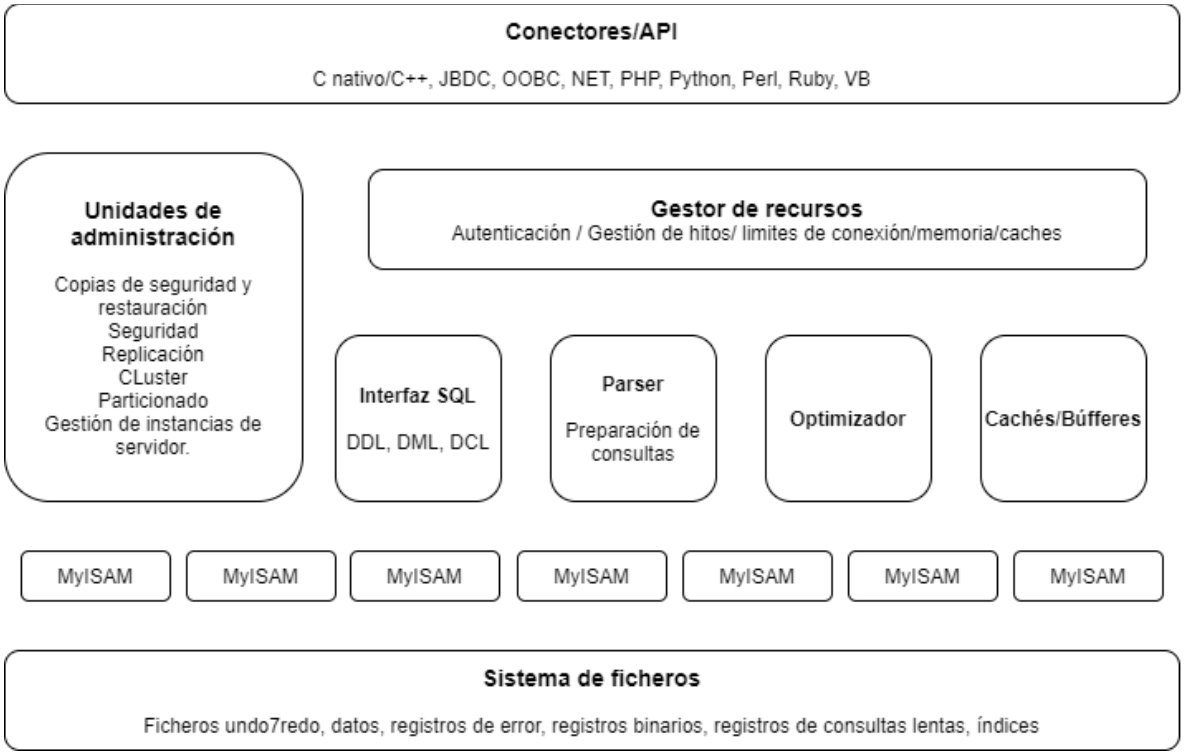


Ilustración 20 Diseño de arquitectura (Huesos Ibáñex, 2015:21)

2) Instrucciones básicas

Cuadro 1. Comandos básicos para el gestor de base de datos MySQL

Comando	**Función**
SHOW DATABASES;	Consulta las bases de datos existentes.
CREATE DATABASE db_nombre;	Crea la base de datos con el nombre que se le asigna.
USE db_nombre;	Definimos que utilizaremos la base de datos con el nombre específico, para realizar consultas u otros.
CREATE TABLE tb_nombre(ATRIBUTO varchar(#));	Creación de la tabla dentro de la base de datos en uso, se asigna el nombre de la tabla, dentro de paréntesis irán los nombres de los atributos y que tipo de datos.
SHOW TABLES;	Para ver las tablas que están dentro de la base de datos.
DROP TABLE tb_nombre;	Elimina la tabla con el nombre específico.
INSERT INTO tb_nombre VALUES (valor);	Inserción de datos en la tabla con el nombre que corresponde, con los valores para cada atributo separados con “,”.
SELECT * FROM tb_nombre;	Consultar los datos almacenados dentro de la tabla ingresando el nombre de este.
SELECT atributo FROM db_nombre;	Consulta de atributos específicos separados por “,” de la tabla.
SELECT atributo FROM db_nombre WHERE atributo = “valor”;	Consulta limitada por el valor que tenga un atributo en específico.

B. CONCEPTOS TÉCNICOS

1. Algoritmos

El algoritmo es una guía desarrollada para realizar una tarea de tal manera que sea ordenada, estructurada, que da solución a un problema, como lo define (Ebratt Gómez, Mancilla Herrera, & Capacho Portilla, 2016:6) Un algoritmo es un conjunto finito de reglas bien definidas en su lógica de control que permite la solución de un problema en una cantidad finita de tiempo. Se emplean los pasos de principio a fin, detallando cada uno de los pasos empleados que ayudará a otras personas a comprender dicha solución. Existirán múltiples

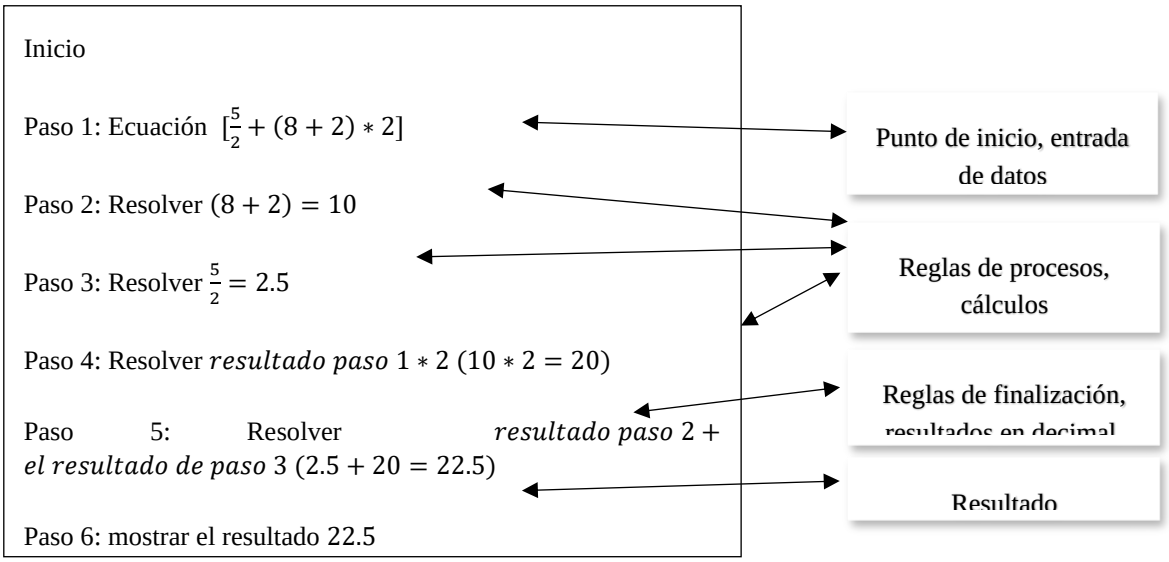
algoritmos para dar solución a un único problema ya que la base fundamental es la lógica. Los algoritmos son empleados en la vida cotidiana que con su uso repetitivo se convierte en hábitos o una acción que se realiza.

Un algoritmo comprende con las características siguientes:

- **Punto de inicio:** se definen las entradas necesarias para ejecutar las reglas o procedimientos que son requeridas para la solución.
- **Conjunto de reglas o procesos:** las reglas que se deben emplear sobre las entradas para analizar, ejecutar y calcular para la solución al problema.
- **Reglas de finalización:** se derivan las reglas para dar con el resultado.
- **Resultado:** define la regla de cómo se presentará los resultados ya sea de manera exacta, valores decimales y otro formato requerido.

Ejemplo de un algoritmo: Algoritmo para la solución de la ecuación $[\frac{5}{2} + (8 + 2) * 2]$

Ilustración 21. Ejemplo de un algoritmo

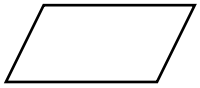
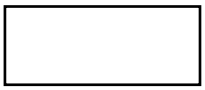
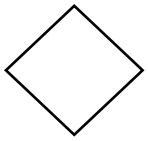
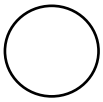
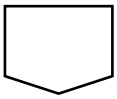
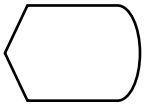


2. Diagrama de flujos

Es un proceso de transcripción de un algoritmo en diagramas, en la diagramación se cuentan con diversas figuras que representan una acción del algoritmo. Es denominada como diagrama de flujo porque se representa el flujo de los procesos mediante una flecha.

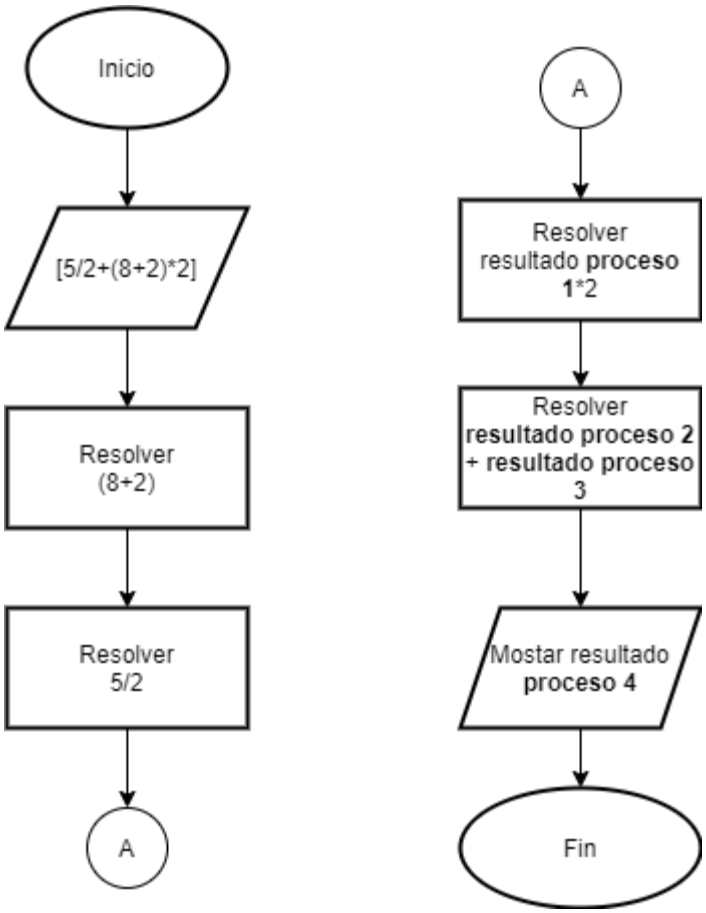
Su principal objetivo es facilitar la comprensión de los flujos de procesos que se están implementando dentro del algoritmo.

Cuadro 2. Figuras y descripción utilizadas en el diagrama de flujo

Símbolo	Descripción
	**Inicio y fin** Es utilizada para definir el inicio del diagrama y como la finalización de este.
	**Entrada o salida** Es utilizada para requerir datos externos o alguna acción que amerite la recepción. Utilizada también comúnmente para salida de datos para dar a conocer alguna información.
	**Proceso** Representa una acción donde se ven involucrados datos o alguna instrucción específica.
	**Decisión** Se utiliza para validar alguna acción y de esta se despliega dos posibles resultados el primero es un verdadero y el último el valor de falso.
	**Conector** Se utiliza para conectar dos puntos del diagrama dentro de la misma página.
	**Conector páginas** Se utiliza para conectar dos puntos del diagrama en diferentes páginas.
	Flujo del programa Representa la dirección que se debe seguir después de haber ejecutado un proceso.
	Pantalla Se utiliza para mostrar una información en pantalla.

Ejemplo de diagrama de flujo del ejemplo $[\frac{5}{2} + (8 + 2) * 2]$ del tema de algoritmos.

Ilustración 22. Ejemplo de diagrama de flujo



3. UML

Es un Lenguaje Unificado de Modelado (UML Unified Modeling Language), que brinda diversos diagramas para representar las ideas de diferentes puntos de vistas, empleado principal mente en realizar, diseñar y documentar el desarrollo de un sistema.

El objetivo de los diagramas es representar la funcionalidad de un sistema, a continuación, se define los tipos de diagramas que son utilizadas en UML.

a. Tipos de diagramas.

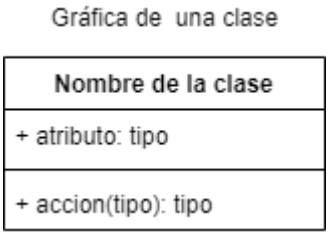
1) Diagrama de clases

Describe la estructura de manera estática de un sistema, presentando la estructura de los parámetros que se pueden implementar.

Una clase es un grupo de cosas que tiene propiedades, similar a lo que pasa con los atributos de la base de datos. Un ejemplo de una clase puede ser: un carro donde pueden contener los atributos color, modelo, motor, transmisión, entre otros y este identifica a un objeto dentro del sistema.

El diagrama de clase se representa con rectángulos dividida en tres partes, en la superior muestra el nombre de la clase, en la mitad los atributos que tiene la clase y por último las acciones que tendrá.

Ilustración 23. Ejemplo de diagrama de clases

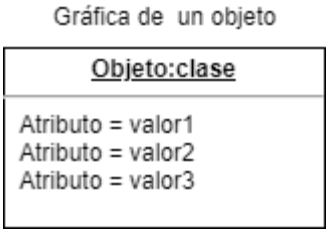


2) Diagrama de objetos

Están vinculadas con los diagramas de clases ya que un objeto es una instancia de una clase ya definida, nos ayuda a determinar la precisión de los diagramas de clases.

El diagrama de objeto se representa con rectángulos dividido en dos, la primera ira el nombre del objeto al cual instancia de la clase y en la parte inferior define un valor específico para el atributo.

Ilustración 24. Ejemplo de diagrama de objetos



3) Diagrama de caso de uso

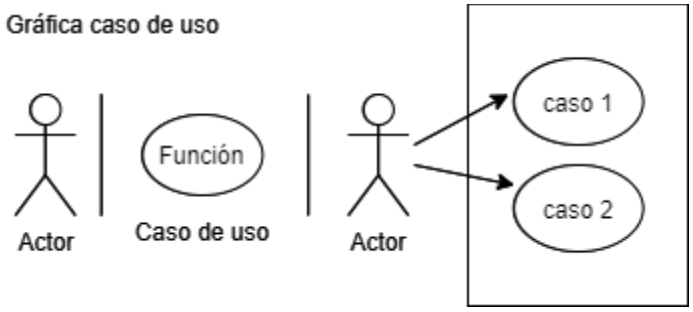
Describe las acciones que se realizan en las secciones de un sistema desde el punto de vista de un usuario, estos modelan la funcionalidad de actores y los casos de uso que se cuenta.

Actor: representa a un usuario de sistema.

Caso de uso: una acción que realiza el sistema.

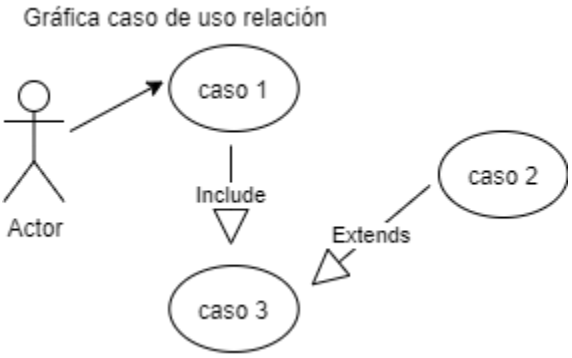
Sistema: donde se ve involucrado el actor con los casos de uso dentro de un rectángulo denominado sistema.

Ilustración 25. Elementos del diagrama de caso de uso



Las relaciones se representan con un conector representados con una acción, Include representa que un caso de uso es necesario para otro y Extends indica que es una opción.

Ilustración 26. Relaciones entre los casos de usos



4) Diagrama de estado

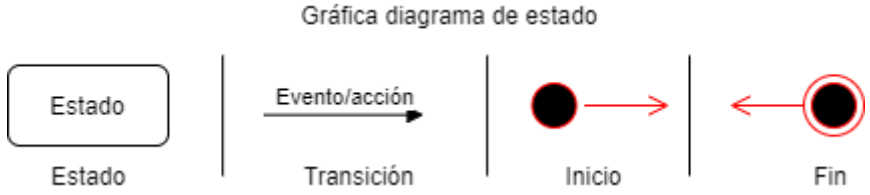
Representa la actividad o inactividad de un objeto del sistema.

El estado: representa la situación durante la vida del objeto.

Transición: define el comportamiento entre los estados de un objeto identificado con la acción que realiza.

Inicio y fin: el comienzo del estado y cómo termina.

Ilustración 27. Elementos del diagrama de estado

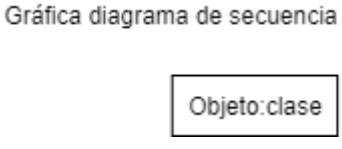


5) Diagramas de secuencias

Los diagramas de clases y objetos están de manera estática en el modelo, pero dentro del sistema estos se interactúan entre sí para determinar comportamientos. El diagrama de secuencia representa esta interacción que se realiza.

Rol de clase: describe el comportamiento de un objeto.

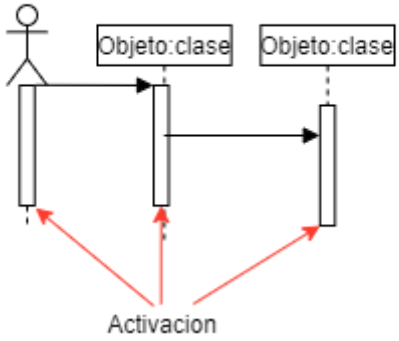
Ilustración 28. Rol de la clase dentro del diagrama de secuencia



Activación: muestra el tiempo de un objeto que emplea para completar.

Ilustración 29. Activaciones dentro del diagrama de secuencia

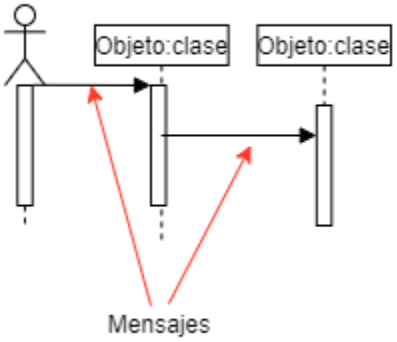
Gráfica diagrama de secuencia / Activaciones



Mensaje: representa la comunicación entre los objetos.

Ilustración 30. Mensaje dentro del diagrama de secuencia

Gráfica diagrama de secuencia / Mensajes

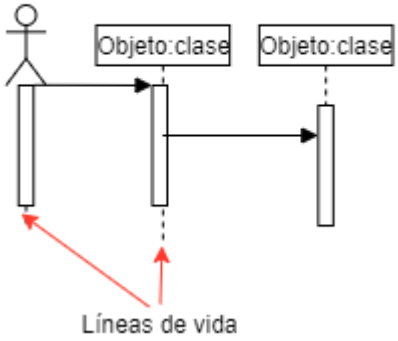


Flecha	Tipo de mensaje
→	Simple
→	Síncrona
→	Asíncrona
↩	Rechazado
→	Time oup

Líneas de vida: indica la presencia del objeto en el tiempo.

Ilustración 31. Líneas de vida dentro del diagrama de secuencia

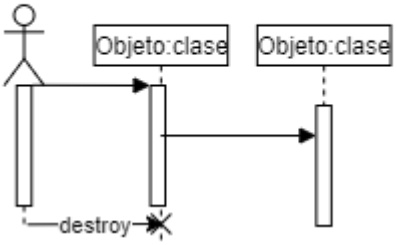
Gráfica diagrama de secuencia / Líneas de vida



Destrucción del objeto: se agrega la etiqueta destroy que apunta en donde termina.

Ilustración 32. Destrucción de un objeto dentro del diagrama de secuencia.

Gráfica diagrama de secuencia / Destruir objeto



6) Diagrama de actividad

Representa la dinámica que tiene un sistema mediante el modelado fijo que ocurre de actividad en actividad, la actividad representa alguna operación de la clase. Se emplean para modelar el flujo de trabajo interno que debe hacer el sistema.

Las figuras que encontramos dentro del diagrama de actividades son las siguientes:

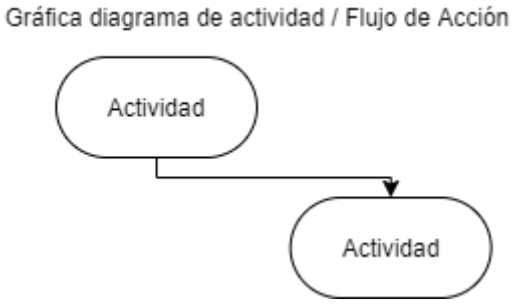
Acción: un estado de acción representa acciones que no deben ser interrumpida de los objetos.

Ilustración 33. Acción de un objeto en el diagrama de actividad

Gráfica diagrama de actividad / Acción

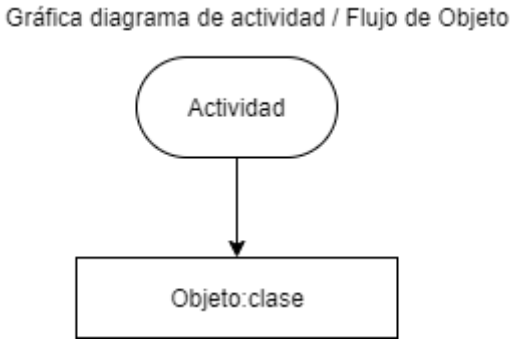
Flujo de acción: determina la relación entre los estados.

Ilustración 34. Flujo de acción dentro del diagrama de actividad



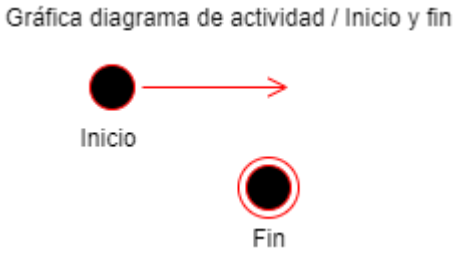
Flujo de objetos: representa el flujo de los objetos en la creación y modificación.

Ilustración 35. Flujo de los objetos con relación a la acción



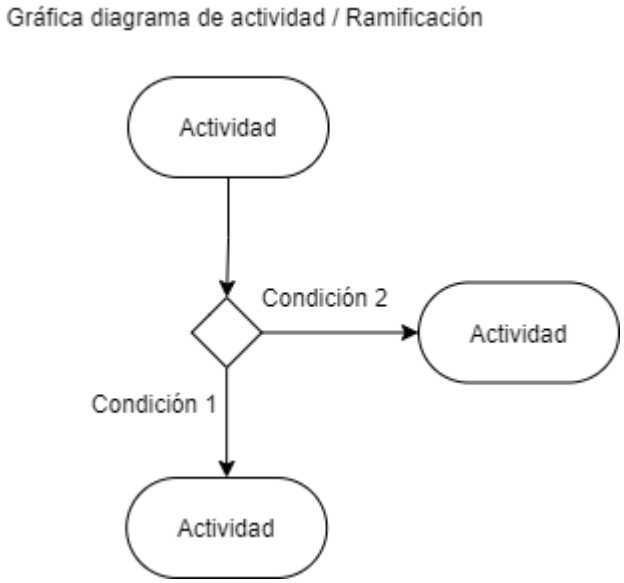
Inicio y final: representa el inicio y fin de un estado de acción.

Ilustración 36. Remarca el inicio o fin del diagrama de actividad



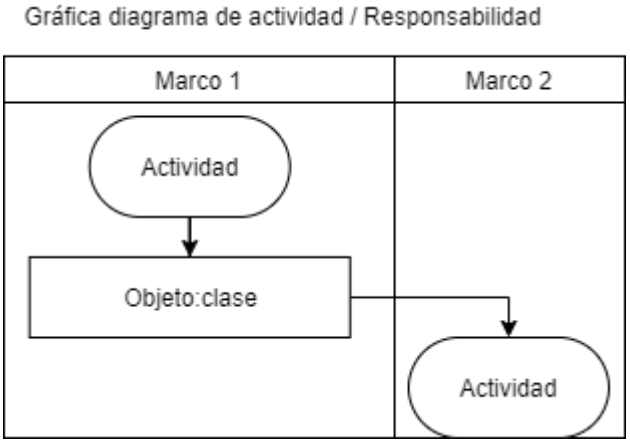
Ramificación: decisión de las salidas, deben estar identificada con el resultado del condición.

Ilustración 37. La ramificación de actividades dentro del diagrama de actividades.



Marcos de responsabilidad: agrupan las actividades relacionadas en una columna.

Ilustración 38. Responsabilidades dentro del diagrama de actividades

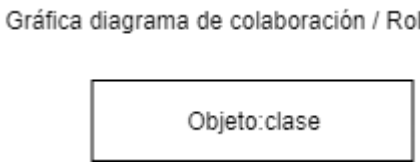


7) Diagrama de colaboración

Se describe las interacciones entre los objetos en términos de mensajes, tomando información de los diagramas de clases, secuencias y caso de uso para describir su comportamiento.

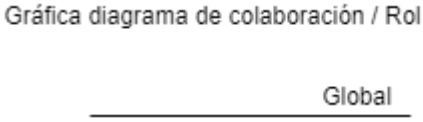
Rol de clase: describe cómo se comporta un objeto y dentro de esta los atributos del objeto no se listan.

Ilustración 39. Rol de clase dentro del diagrama de colaboración



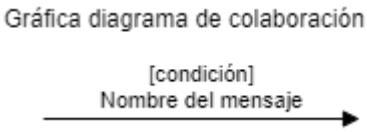
Rol de asociación: describe el comportamiento de una asociación.

Ilustración 40. Rol de asociación dentro del diagrama de colaboración



Mensajes: se enumeran los mensajes según el orden de ejecución.

Ilustración 41. Mensajes dentro del diagrama de colaboración



4. Estructura cliente servidor.

Es un modelo para el desarrollo de software diseñados en compartir los recursos a de un servidor dentro de una red local, WAN u otro que sea requerida por la magnitud del problema que da solución el software. Los clientes que interactuarán con el sistema deberán estar conectados a la estructura de la red para poder acceder a los recursos como lo serían la base de datos, registros, archivos y otros.

a. Sistemas de servidor

Es el sistema desarrollado para cumplir con una tarea específica, dichos sistema es el encargado de gestionar las peticiones de los clientes, de manera que este sistema debe estar en ejecución el tiempo necesario.

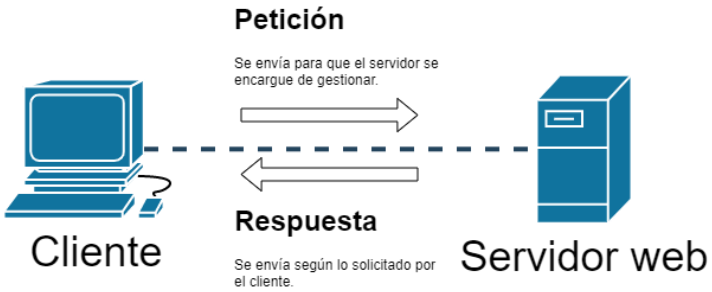
b. Sistemas de clientes

Para que el sistema pueda ser utilizada por los clientes es necesaria de la intervención de un navegador web siendo estas, Opera, Mozilla, Chrome o Microsoft Edge para acceder a la ruta o dirección al cual se da acceso al sistema alojada dentro del servidor.

c. Funcionamientos

El cliente entra en el sistema cliente para poder realizar alguna acción correspondiente, en el sistema servidor se encarga de gestionar las peticiones realizadas por el cliente, una inserción, modificación, consulta, eliminar registro o delegar alguna funcionalidad necesaria.

Ilustración 42. Funcionamiento estructura cliente servidor



El ejemplo de un sistema de banco: el usuario ingresa a la dirección web, al momento de ingresar se envía una petición al servidor del banco y regresa la respuesta accediendo hacia la página principal, este proceso es repetitivo en la funcionalidad del sistema.

5. Lenguaje de marcas HTML.

Es un lenguaje especial para el desarrollo de páginas webs, consta de etiquetas o (tags) que define los elementos que irán dentro de la página web. Es denominado lenguaje de enmarcado ya que se marcan con las etiquetas para dar una funcionalidad específica dentro de la página web.

a. Etiquetas o tags

Estas definen la estructura de la página como lo son las cabeceras, el cuerpo de la página y como un pie de página, la forma de escribir una etiqueta se define con el símbolo menor que <, seguido del nombre de la etiqueta y para luego terminar con el símbolo mayor que >; ejemplo: <etiqueta>.

Al utilizar las etiquetas estas se deben inicializar y como finalizar, para finalizar una etiqueta se debe colocar el símbolo diagonal / al comienzo del nombre de la etiqueta; ejemplo <etiqueta>Contenido</etiqueta>.

Algunas etiquetas se pueden cerrar en el inicializador del mismo, ejemplo <etiqueta />.

Alguno de estos también se puede utilizar si necesidad de algún cierre de etiquetas, ejemplo <etiqueta>.

1) Atributos

Los atributos de los elementos identifican características especiales que pueden tener, la forma de escribir se define por el nombre del atributo, luego el símbolo igual = y para luego agregar el valor entre comillas o doble comillas ejemplo atributo=”valor”.

Cuadro 3. Etiquetas más comunes para el lenguaje de enmarcado HTML

Nombre de etiqueta	**Función**	**Atributos**	**Función**
Base			
<html></html>	Muestra en la página todo lo que esté dentro de la etiqueta.	Lang	Define el idioma
Metadatos			
<head></head>	Define el encabezado de la página, donde contendrá recursos y configuraciones.		
<meta>	Configuración para los buscadores y de la página web.	name content charset	El nombre de la web. Una descripción de la web. Formato de caracteres utilizados.
<style></style>	Reglas de estilos para el documento.		
Secciones			
<body></body>	Define el cuerpo de la página.		
<header></header>	Define una cabecera.		
<section><section>	Una sección de la página web.		
<nav></nav>	Área de navegación o menú.		
<footer></footer>	Define un pie de página.		
Bloques de contenido			
<main></main>	Un bloque principal.		
	Una división de un área específica.		

Nombre de etiqueta	Función	Atributos	Función
<hr>	Separador.		
Texto			
 	Salto de línea.		
<a>	Establece un hipervínculo.	href	Establece la dirección al cual será vinculada.
Formularios			
<form></form>	Establece el área del formulario.	action method	Define la ruta que atenderá la petición. Método de envió de la petición (GET/POST)
<input/>	Define una entrada de dato.	type name value	De tipo text, password, email, file, img, number, entre otros. Un nombre único para ser interpretado el valor que lleva. Un valor específico.
<option></option>	Una caja de lista de opciones.	selected Value	El valor seleccionado por defecto. El valor que tiene la opción a seleccionar.
<textarea></textarea>	Un área de texto.	name value	Un nombre único para ser interpretado el valor que lleva. Un valor específico.

Nombre de etiqueta	Función	Atributos	Función
Tabla			
<table></table>	Establece el área de la tabla.	border	Borde de la tabla.
<thead></thead>	Cabecera de la tabla.		
<tbody></tbody>	Cuerpo de la tabla.		
<tr></tr>	Fila de la tabla.		
<td></td>	Una celda.		
Otros			
<img/>	Define la inserción de una imagen.	src width height	La ruta donde esta almacenada la imagen. Ancho. Alto.
<lu></lu>	Una lista no ordenada		
	Elementos de la lista		

b. Estructura de la página web

1) Inicializador

Define el inicio del lenguaje enmarcado y se define de la siguiente manera <html> contenido de la página web </html> y todo lo que contenga esta etiqueta será visible en la página web.

2) Encabezado

La sección donde se definen la configuración de la página web, definiendo los componentes necesarios que serán utilizados dentro de la página web, definido de la siguiente <head> todas las configuraciones y componentes </head>

3) Cabecera

Una sección donde se agregan los elementos más importantes como lo son el logo de la empresa, buscador y un navegador para acceder a otras áreas de la página, definido de la siguiente manera <header>Elementos mencionados anteriormente</header>.

4) Cuerpo

En el cuerpo de la página web se definen los elementos que llevará la página como un menú, títulos, contenido, imágenes, formularios entre otros, que ayudará a mostrar o recolectar información. Definida de la siguiente manera <body>todo el contenido de la página web</bady>

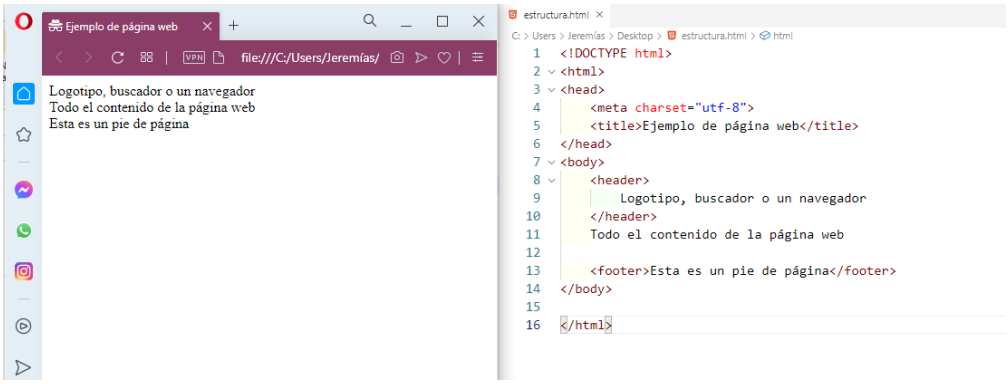
5) Contenido

Dentro del contenido podemos colocar divisiones para la página web, secciones, tamaños de texto, estilo de texto y multimedia que se puede agregar.

6) Pie de página

Conocida como footer, donde se agregan enlaces de navegación o sugerencias para que puedan acceder de manera rápida a estos, definida con <footer>El contenido del pie de página </footer>

Ilustración 43. Ejemplo de la estructura de la página web



6. Diseño Responsive

Un diseño web responsive corresponde a la adaptabilidad de los elementos de una página web a diversas resoluciones de pantallas de los diferentes tipos de dispositivos existente, de tal manera que se puede visualizar desde una computadora hasta un teléfono inteligente. Cumple con el objetivo de proporcionar la información de manera adecuada a cada uno de los clientes que puedan acceder a una página web.

7. CSS.

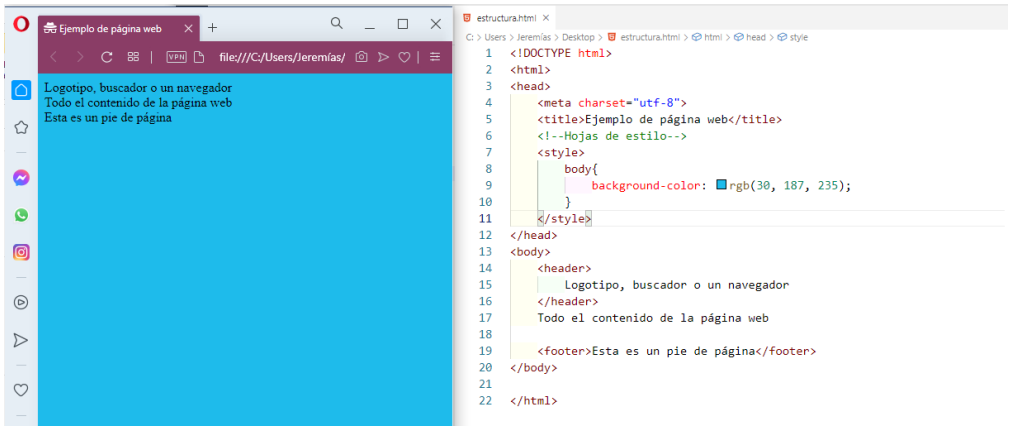
Denominadas hojas de estilo en cascadas, Cascading Styke Sheet CSS que es un lenguaje de diseño, que nos ayuda a mejorar el diseño de la página web mediante el uso de colores, posicionamientos, borde, fondos, punteros entre otros más, ayuda a mejorar el diseño del interfaz que vera el usuario de la página web.

Para su integración dentro de la estructura HTML, se define dentro de la etiqueta head, mediante la etiqueta específica style, <style>Todo el diseño necesario</style> y dentro de estos se colocan la sintaxis para dar un diseño denominadas reglas. También es posible colocar dentro de las etiquetas las reglas específica cobre ese elementos de esta manera, <etiqueta style=”regla”/>.

a. Selectores

Un selector define a que etiqueta de la estructura del HTML se le aplicará una serie de reglas. Los selectores también pueden definir a los atributos id y class, para poder aplicar las reglas a todas las etiquetas que contengan este atributo.

Ilustración 44. Ejemplo de selectores con reglas definidas



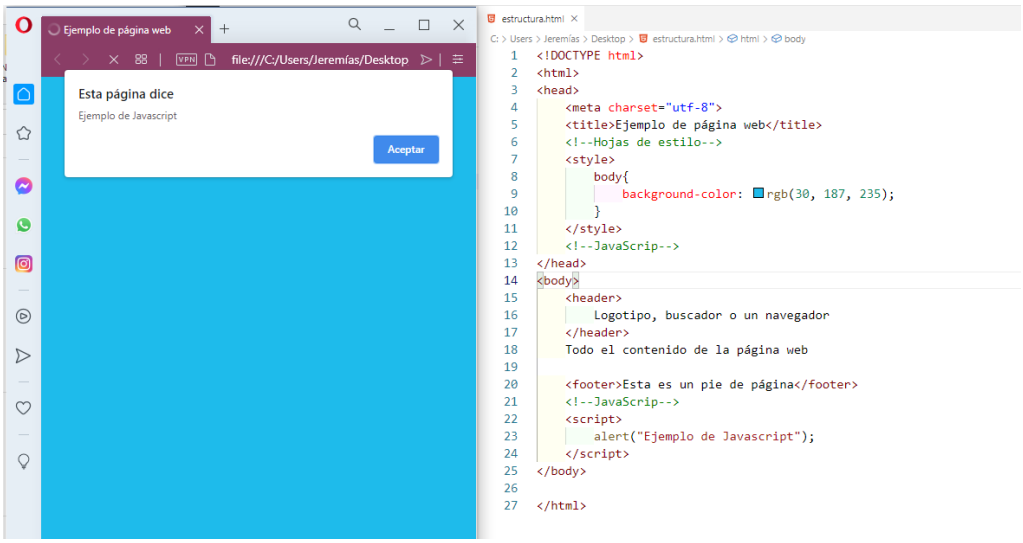
8. JavaScript.

Es un lenguaje desarrollado para que sea ejecutado por el navegador conocida también como lenguaje del lado del cliente, utilizado para que las páginas web sean dinámicos, únicos e interactivos con el usuario y se define dentro de la etiqueta head, como también dentro de la etiqueta body y como una regla siempre se define antes del cierre de esta etiqueta, se empieza con le etiqueta <script> código </script>.

Algunas funcionalidades:

- Al momento de dar clic a un botón podemos hacer una acción.
- Validar un valor ingresado dentro de un input.
- Agregar clases a las etiquetas
- Eliminar secciones de la página web
- Crear elementos para las etiquetas
- Agregar reglas de CSS

Ilustración 45. Agregando un script dentro de HTML



a. Librería JQuery

Es una librería que ayuda a los desarrolladores web a escribir menos código de lo que se haría con JavaScript, ayuda a obtener una estructura más limpia y a reducir el tiempo en su escritura.

b. Librería Vibrant

Es una librería desarrollada por el usuario jariz de GitHub que nos ayuda a recorrer los colores de una imagen píxel por píxel previamente cargadas dentro de un input tipo archivo, para tomar un grupo de muestra de colores en formato hexadecimal o RGB. Mediante las funciones escritas dentro de esta librería se puede hacer uso de varias opciones como obtener el valor más repetitivo, obtener el nombre del color y el color; con estas funciones nos retorna un array de datos para luego poder utilizar a nuestro criterio.

c. Librería Intense Viewer

Es una librería desarrollada por el usuario tholman de GitHub que nos ayuda a realizar zoom a las imágenes dentro de nuestra página web, mediante su función establecemos los elementos img que tomara todas las etiquetas img para aplicar el zoom respectivo.

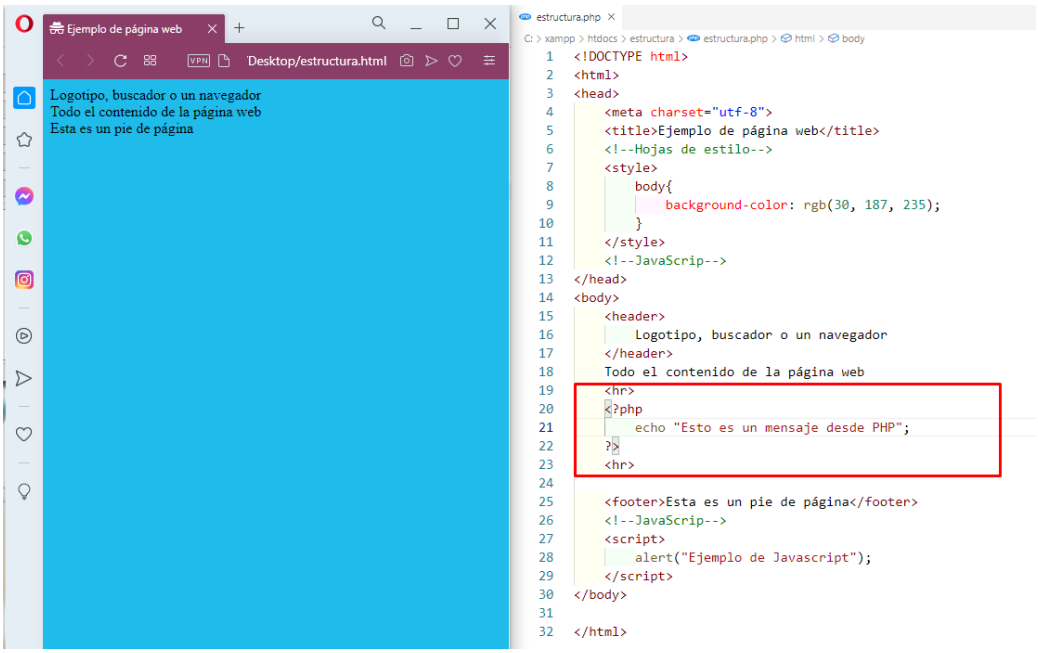
9. PHP.

Es un lenguaje interpretado del lado del servidor, que ayuda especialmente el manejo de las peticiones que son realizadas por los usuarios, manejo de los datos, comunicación a un gestor de base de datos. Para poder utilizar este lenguaje es necesario instalar el intérprete del lenguaje dentro el servidor web que será utilizado para resolver las peticiones. Es importante que los archivos que sean interpretados tengan la extensión .php.

a. Sintaxis

¿Para implementar este lenguaje dentro de la estructura HTML, es necesario delimitar el código mediante <?php código php?> dentro de estos delimitadores agregamos las sintaxis que pueden ser mostrar texto, variable, ciclos, condiciones entre otros.

Ilustración 46. Insertar código PHP en THML



10. Laravel 8.x.

Laravel es un Framework diseñada para el lenguaje de programación PHP para manejar el código de manera limpia y sencilla, evitando generar mucho código ya que éste cuenta con códigos PHP prescritos como la conexión a la base de datos, imprimir variables, administrador de autenticación propia entre otros más.

a. ¿Por qué Laravel?

Laravel nos ofrece una escalabilidad, progresividad y una comunidad, que nos brinda soporte para el desarrollo de nuestros sistemas web.

Escalabilidad: Ofreciendo al proyecto diversos recursos que ayudan a escalar según la evolución que tenga el sistema.

Progreso: ya que con la documentación y recursos que se tiene hace que el crecimiento en su uso sea progresivo.

Comunidad: mediante la combinación de los recursos desarrollados por la comunidad han ayudado al desarrollo de este Framework.

b. Características de Laravel

Se basa en el principio del modelo MVC (Modelo, vista, controlador), que nos ayuda en el flujo del sistema que será desarrollada. Cuenta con un motor de plantillas denominada Blade, que nos brinda más agilidad en el desarrollo al reutilizar códigos.

Eloquent ORM, se utilizan para realizar peticiones a la base de datos del proyecto mediante PHP. Base de datos y migraciones, ayuda en el manejo de la base de datos en producción ya que estos se pueden modificar si necesidad de eliminar y crear de nuevo.

Ejecuta las tareas de manera asíncrona para mejorar el rendimiento del sistema.

1) Estructura de Laravel

a) Modelo

Es el que administra las peticiones hacia la base de datos, cada modelo administra una tabla específica de la base de datos donde se puede realizar las inserciones, consultas, actualización y eliminar registros de la tabla que el modelo esté administrando.

b) Vista

Para la vista se implementa un sistema propio de Laravel denominada Blade.

Blade: es un sistema de plantillas integrada dentro de Laravel que nos ayuda a reutilizar códigos HTML.

c) Controlador

Es quien controla toda la lógica de la programación, como sería las consultas necesarias, envió de variables a las vistas y como gestiona las rutas del sistema.

d) Rutas

Administra las peticiones de los usuarios y se encarga de enviar hacía el destino correspondiente.

11. Modelo de desarrollo de prototipo.

Este modelo forma parte de los modelos de desarrollo de software que su función es entregar un posible resultado en menor tiempo posible, la función del modelo es presentar un interfaz, entradas y salidas que son necesario para la solución a un problema específico. Dentro de este proceso el cliente está involucrado de manera directa, ya que ira evaluando cada módulo o funciones, para su modificación o validación y este proceso es repetitivo hasta conseguir el resultado esperado por el cliente.

Algunas de las características que podemos describir del modelo son los siguientes:

- Un sistema funcional que dé solución al problema
- Valida las posibles soluciones de acuerdo con las necesidades del problema.
- Su coste es bajo respecto a los demás modelos.
- Se puede realizar en poco tiempo.

a. Etapas del modelo

1) Recopilación y refinamiento de requisitos

Recopilación de información: es el proceso donde se aplican metodologías de investigación como lo son las encuestas, observaciones, cuestionario y todo aquello necesario en la recopilación de la información. La información que se debe recopilar debe estar basados en tres aspectos importante que son las entradas, los procesos que son ejecutadas de ser necesario fórmulas y como último las salidas esperadas.

Refinamiento: es el proceso donde se toman únicamente la información necesaria, se debe agrupar o segmentar cada uno de estos.

2) Modelado diseño rápido

El objetivo de esta fase es determinar los requerimientos del software de acuerdo con los estándares del cliente, deseos y funcionalidades que se quiere implementar. Dentro de esta se determinan también las especificaciones necesarias para el desarrollo del software.

3) Construcción del prototipo

En esta fase se realiza los diseños de interfaz, algoritmos, diagramas de flujo, diagrama entidad relación entre otros que sean necesario dentro del desarrollo del software. Se determina las herramientas de desarrollo como el tipo de servidor con sus características y el lenguaje donde será codificada el sistema.

4) Desarrollo

Dentro de la fase de desarrollo se toman cada uno de los diseños definidos en la fase anterior para empezar a modelar cada una de estas, de acuerdo con las sintaxis de los lenguajes a utilizar.

Se define cada proceso que se deben realizar desde los componentes físicos hasta los componentes lógicos a implementar.

5) Evaluación del prototipo

Esta fase es una de las bases que determina si lo desarrollado cumple con los requisitos dados por los clientes, lo que se espera es la aprobación, de ser necesario modificar las partes que no son funcionales o no validadas por el cliente.

6) Refinamiento del prototipo

Se deben emplear pruebas para evaluar el sistema, estos pueden ser pruebas de estrés, ya que de esta manera podremos observar si el sistema cumple con lo requerido.

12. Herramienta de desarrollo.

a. Visual Studio Code

Es un editor de código desarrollado por la empresa de Microsoft, es un software multiplataforma que se pueden instalar en Windows, Linux y Mac OS. Es una herramienta que ayuda a los desarrolladores de software en el resaltado de sintaxis en los siguientes lenguajes Python, PHP, CSS, HTML, JSON, R, JavaScript, Java y entre otros más, incorpora un autocompletado para los siguientes lenguajes JavaScript, HTML, CSS, JSON, TypeScript, Less y Sass. Cuenta también con un depurador de código para los lenguajes de programación como JavaScript, C#, C, C++, Python y PHP, esto ayuda a los desarrolladores a compilar los códigos dentro del editor sin necesidad de instalar otras herramientas.

1) ¿Por qué visual estudio?

Cuenta con múltiples extensiones que ayudan a mejorar el flujo de la codificación para diversos lenguajes. Dispone de una terminal incorporada para ejecutar comandos dentro del origen del proyecto. El diseño de su interfaz es intuitivo para los desarrolladores.

b. XAMPP

Es un software distribuido por Apache, donde incluye el propio Apache que es un servidor web, el gestor de base de datos MySQL, el intérprete de PHP y PERL.

El software tiene como función simular un servidor web donde es instalada, pensada para realizar pruebas ya que si se emplea un sistema con todas sus capas es muy costoso.

1) ¿Por qué XAMPP?

Es uno de los software más utilizadas por la comunidad de desarrolladores.

Cuenta con un administrador de base de datos denominada phpMyAdmin que facilita la visualización de las bases de datos y su contenido.

c. Composer

Es un sistema que gestiona los paquetes de PHP, donde incluyen las librerías y dependencias del mismo lenguaje para aplicarlos a los proyectos, cada una de estas pueden ser realizadas desde la terminar empleando instrucciones específicas.

1) ¿Por qué Composer?

Ya que este gestor de paquetes es fundamental para poder instalar las librerías del lenguaje de programación PHP.

Desde el gestor de paquete podemos gestionar todas las librerías y limitar las características.

2) Comandos

Require: especifica las características de las librerías que necesitamos dentro de nuestro proyecto, dentro de un archivo denominada composer.json se almacena las especificaciones de la librería, la versión mínima y máxima.

Install: instala las dependencias requeridas dentro del archivo composer.json.

Update: actualiza los parámetros de las dependencias que serán utilizadas predefinidas dentro de composer.json.

Remove: desinstala las librerías y los elimina dentro del registro de composer.json

13. Bootstrap.

Es una biblioteca desarrollada para creación de diseños de interfaz de páginas web responsive empleando CSS y JavaScript a través de las etiquetas de HTML utilizando los atributos y clases.

Su funcionamiento se basa en reglas de estilos ya escritas almacenadas dentro de un fichero con el nombre de bootstrap.main.css, donde están las reglas escritas. Las funciones de JavaScript esta almacenadas dentro del fichero bootstrap.main.js, en este están codificadas funciones que se utilizan para interactuar con el usuario de mejor manera, un ejemplo de esto son los modales y como el Carousel que emplea funciones de JavaScript.

1) ¿Por qué Bootstrap?

Es una de las grandes librerías que cuenta con una documentación extensa sobre los componentes que dispone, esto ayuda a que la integración para los proyectos sea fácil.

Al ser una de las librerías más utilizadas por la comunidad de desarrolladores cuenta con múltiples foros donde se pueden consultar dudas sobre su integración.

V. MARCO METODOLÓGICO

En el desarrollo del sistema para la tienda de ropa típica, se empleó el modelo de prototipos evolutivo que ayudó en la recopilación de los requerimientos necesario para el desarrollo de interfaz y de la lógica de los algoritmos para cumplir con los objetivos especificados. La importancia de haber completado estos procedimientos fueron la clave para dar solución a las necesidades del cliente.

A. PLANIFICACIÓN DEL PROYECTO

Se emplearon un total de 8 semanas para el desarrollo del sistema web, teniendo como base el modelo de desarrollo de prototipos, distribuidos de la siguiente manera:

- **Semana uno:** realización de observación y una entrevista la dueña de la tienda.
- **Semana dos:** con la información recopilada, se modelo los requerimientos del sistema y los casos de uso que tendría en sistema.
- **Semana tres:** ya con los modelos definidos, se maquetó el diseño del interfaz de usuario y las funcionalidades lógicas del sistema.
- **Semana cuatro:** con la maqueta realizada, se comienzo del desarrollo del sistema.
- **Semana cinco:** se realizó visita con el cliente para determinar el refinamiento de los datos basados en la información del producto.
- **Semana seis:** se realizó las modificaciones mencionadas por el cliente y agregando la funcionalidad de enviar los datos hacia el servidor.
- **Semana siete:** se presentó la funcionalidad de referencia al cliente, teniendo en cuenta que se realizó algunos registros dentro del sistema como parte de las pruebas respectivas. En el último día de la semana se presentó el prototipo final al cliente.
- **Semana ocho:** se realizó la documentación respectiva, dejando evidencia el código y la funcionalidad de cada uno.

ACTIVIDAD	INICIO DEL PLAN SEMANA	DURACIÓN DEL PLAN SEMANA	SEMANAS 8/03/2021							
			1	2	3	4	5	6	7	8
Fase 1: Recopilación y refinamiento de requisitos	1	1								
Fase 2: Modelado diseño rápido	2	1								
Fase 3: Construcción del prototipo	3	1								
Fase 4: Desarrollo	4	4								
Fase 5: Evaluación del prototipo por el cliente	5	2								
Fase 6: Refinamiento del prototipo	6	2								
Documentación	1	8								

B. RECOPILACIÓN Y REFINAMIENTO DE REQUISITOS

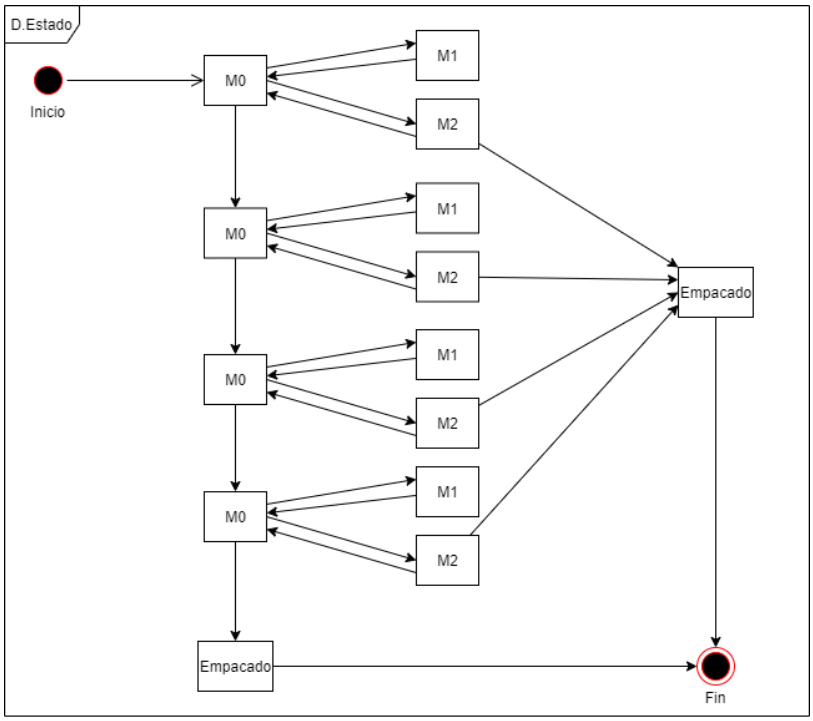
1. Tienda de ropas típicas

Es una tienda ubicada en la comunidad de Aldea la Esperanza Totonicapán, Totonicapán; ofreciendo a la venta ropa típica, ropa para damas, caballeros y niños. Teniendo como base la venta de ropa típica para damas, ofrece sus servicios desde hace más de diez años, es una de las tiendas más grandes y modernas en la localidad, teniendo clientes fieles por la calidad de sus productos.

2. Análisis de proceso de la venta de ropa típica

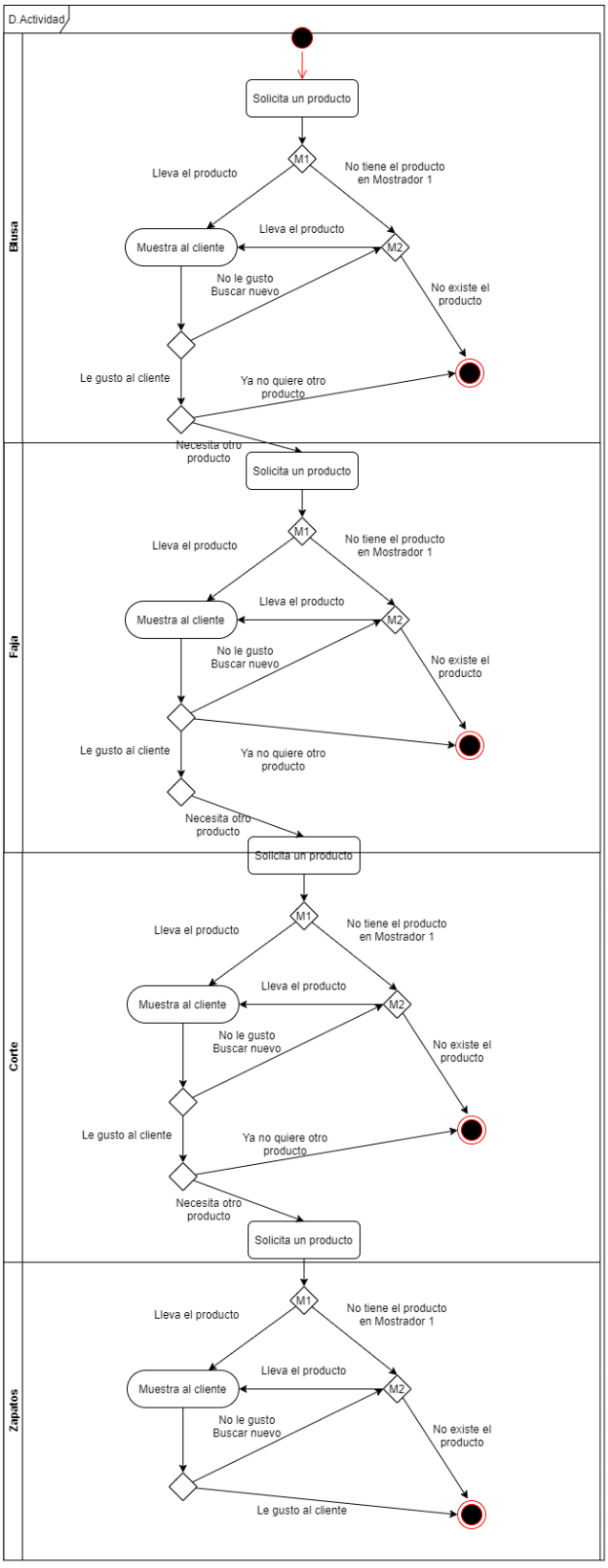
Para analizar el proceso que se debía realizar el cliente fue necesario aplicar la metodología de investigación observación, ya que cada actividad que realizaban no se podía medir de otra manera. El cliente llega a la tienda, pasa ser atendido por un empleada en mostrador principal (M0), el cliente pidió un producto de un color en especial, la cual la empleada fue a buscar entre las que se guardaba en otro mostrador (M1), llevó el producto solicitado por el cliente hacia el mostrador principal (M0), este producto no complacía las necesidades del cliente por tal razón la empleada fue a buscar otro de diferente estilo en otro sitio (M2) regresando hacia el mostrador principal, realizando este proceso por cada producto que es necesario mostrar a cliente.

Ilustración 47. Diagrama de estado del proceso de atención al cliente.



Se logró observar que se realiza este mismo proceso a los cuatro productos de la siguiente manera, un promedio de tres veces con el producto blusa, seis veces en caso del corte, faja un promedio de tres y para los zapatos un promedio de tres veces.

Ilustración 48. Diagrama de actividad del proceso de atención al cliente



3. Resumen

Uno de los factores principales que se logró identificar es la falta de visualización de los productos hacia sus clientes y esto hace que los empleados deben buscar y presentar al cliente, tengamos en cuenta que no disponían de ninguna herramienta que ayudara con la gestión de estos.

Los impactos que esto presentaba a la tienda son los siguientes: el descontento de los clientes ya que le corresponde esperar para ser atendidos, desorden de los productos después de haber atendido a un cliente y como también el descontento del cliente atendido.

4. Información para el sistema

Después de la observación se realizó la recopilación de información para buscar una solución óptima hacia el problema encontrada, haciendo mención que las características que los clientes que tomaban en cuenta son los siguientes:

Cuadro 4. En la figura se presenta cada una de las características del cual se maneja de cada producto que se tiene dentro de la tienda

Blusa	**Faja**	**Corte**	**Zapatos**
Nombre	Nombre	Nombre	Tamaño
Material	Tipo	Estilo	Estilo
Tamaño	Tamaño	Clase	Clase
Precio	Precio	Precio	Precio
Costura	Costura	Costura	
Ubicación	Ubicación	Ubicación	
Clase			

5. Refinamiento de los datos.

Después de haber recabado la información se evaluó las características, dejando agrupado algunos de estos que tenían relación entre sí.

Cuadro 5. Datos refinados

Característica	**Descripción**	**Producto Blusa**	**Producto Faja**	**Producto Corte**	**Producto Zapatos**

Nombre	Un nombre único como un código que diferencie a los demás productos.	X	X	X	X
Estilo	Presenta las figuras y tipo de material.	X	X	X	X
Traje típico	Si es un traje típico de Guatemala.	X	X	X	X
Clase	Existe tres clases siendo la primera el más costoso y de mejor calidad.	X	X	X	X
Estilo de costura	Existen dos los que están elaborados a mano y las computarizas.	X	X	X	X
Tamaño	Tamaño del producto.	X	X	X	X
Precio	Precio del producto	X	X	X	X

C. MODELADO DISEÑO RÁPIDO

Con la información ya refinada se prosiguió a determinar los requerimientos del sistema.

1. Requerimientos funcionales.

Para los requerimientos funcionales del sistema se presentan las siguientes:

Cuadro 6. Requerimientos del sistema web a desarrollar

R_No.	**Nombre**	**Descripción**	**Involucrados**
1	Ingreso al sistema.	Página principal donde se muestra una descripción de la tienda y los productos que ofrece.	Tienda y clientes.
2	Ingresar al módulo de administración.	Deberán ingresar con credenciales, usuario y contraseña.	Tienda.

3	Cambiar contraseña.	El sistema deberá permitir que se pueda actualizar la contraseña del usuario.	Tienda.
4	Creación de usuario.	El sistema podrá crear usuarios que puedan acceder al módulo de administración.	Tienda.
5	Modificar usuario.	Podrán modificar los datos de usuario.	Tienda.
6	Consultar usuarios.	El sistema presentará todos los usuarios registrados.	Tienda.
7	Registro de producto.	El sistema podrá registrar los productos como, blusas, fajas, corte y zapatos.	Tienda.
8	Muestra de colores.	El sistema debe tomar 5 muestras de color del producto.	Tienda.
9	Modificar el producto.	El sistema permitirá la modificación de los datos de un producto.	Tienda.
10	Mostrar los productos.	Mostrar los productos con sus características.	Tienda y cliente.
11	Menú de navegación.	Contará con un menú para la movilidad dentro del sistema.	Tienda y cliente.
12	Contará con un asistente.	El sistema deberá contener un asistente que ayude a los clientes a encontrar el producto que necesitan.	Cliente.
13	Contará con un catálogo.	El sistema mostrará un catálogo de los productos por categoría.	Cliente.
14	Referirá productos.	El sistema deberá referir un producto de acuerdo con el color del producto seleccionado.	Cliente.
15	Traje.	El sistema debe mostrar todas las coincidencias en un formato especial donde se pueda ver de mejor manera la combinación de los productos.	Cliente.

2. Requerimiento no funcional del sistema.

El sistema deberá comprender con las siguientes especificaciones para el interfaz de usuario.

a. Interfaz del sistema

Debe ser un diseño sencillo, entendible para el usuario, empleando colores atractivos y de manejo fácil.

b. Consistencia

Siendo una implementación nueva dentro de la tienda, ésta deberá contener diseño representativo para que los clientes puedan interpretar los componentes del sistema.

c. Requerimiento de hardware

Los requisitos mínimos para el servidor son los siguientes, teniendo en cuenta el tamaño de la tienda.

Cuadro 7. Requerimientos del hardware del servidor

Sistema operativo	Windows 10 o Ubuntu.
Procesador	Procesador Intel® Celeron® 847, 1.10 GHz.
Almacenamiento	10 GB.
RAM	2 GB.
Componentes	Monitor, teclado, ratón, entrada de red RJ45, Salida de video y un UPS.

Los requisitos mínimos para el equipo que se utilizará para el cliente.

Cuadro 8. Requerimiento del hardware del cliente

Sistema operativo	Windows 10.
Procesador	Procesador Intel® Celeron® 847, 1.10 GHz.
Almacenamiento	10 GB.
RAM	2 GB.
Componentes	Monitor, teclado, ratón, entrada de red RJ45, Salida de video y un UPS.

3. Reglas de uso.

a. Seguridad de acceso

Para la conectividad deberán emplear una red local, para el ingreso del módulo de administración únicamente los empleado y propietario de la tienda podrán ingresar mediante un usuario y contraseña.

b. Ingreso de información

Únicamente los propietarios y como empleados podrán modificar los datos del producto y los usuarios en el sistema.

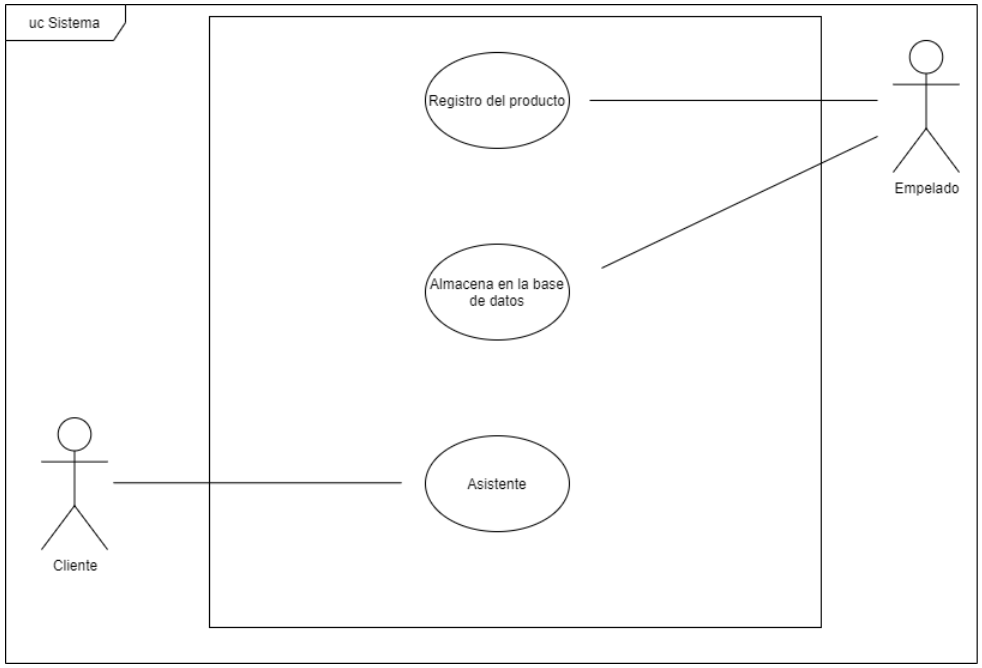
c. Limitaciones del sistema

El sistema se desarrollará en el lenguaje de programación PHP con el framework Laravel y el gestor de base de datos MySQL, empleando herramientas de desarrollo como editor de código Visual Studio Code, XAMPP para el servidor y phpMyAdmin para la manipulación de la base de datos.

4. Caso de uso del sistema.

Describimos el caso de uso del sistema que será utilizada tanto por los propietarios y los clientes.

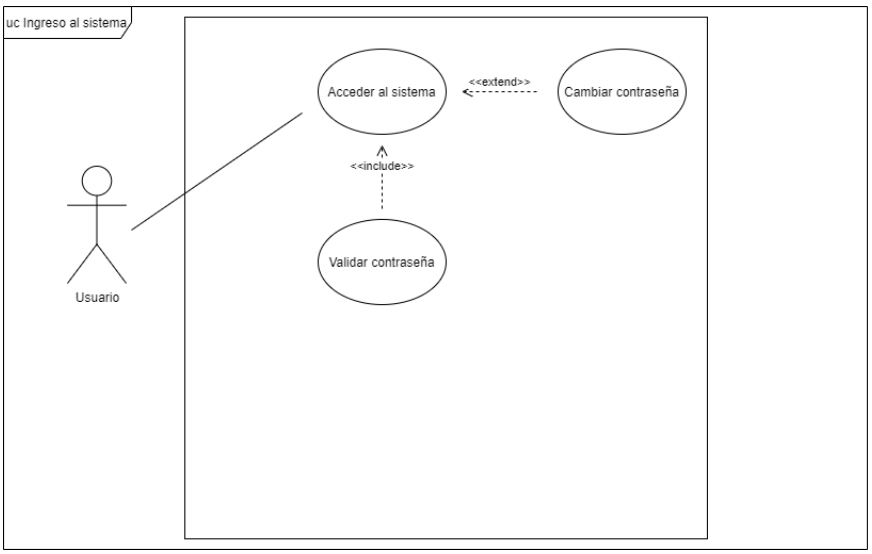
Ilustración 49. Diagrama de caso de uso del sistema web



5. Caso de uso ingreso al módulo de administración.

Define el acceso al módulo de administración para el usuario.

Ilustración 50. Diagrama de caso de uso ingresar al módulo administración.



Del cuadro 9 al 11 se describen las funcionalidades de cada caso de uso.

Cuadro 9. Descripción de caso de uso acceder al sistema

Id caso de uso:	IG-1		
Nombre:	Ingresar al sistema		
Proyecto:	Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.		
Autor:	Santos Jeremías García Tzul	**Versión:**	1.0.0.0
Prioridad:	Media	**Código de requerimiento:**	R_2

Objetivo en contexto:	Ingreso al sistema con usuario y contraseña.		
Actores:	Usuarios		
Entradas:	Las credenciales que son el usuario y contraseña.		
Salidas:	La validación para acceder al sistema.		
Pre-condiciones:	Estar registrado como usuario en el sistema.		
Post-condiciones:	**Final de éxito:**	Ingreso al sistema	
	Final de fallo:	Intentar de nuevo.	

Flujo básico de éxito			
No.	**Actor**	**No**	**Sistema**
1	Usuario ingresa sus credenciales.		
2	Habilita la opción de mantener activo		
3	Activa función de ingresar al sistema		

Cuadro 10. Descripción de caso de uso cambiar contraseña

Id caso de uso:	IG-2		
Nombre:	Cambiar contraseña		
Proyecto:	Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.		
Autor:	Santos Jeremías García Tzul	**Versión:**	1.0.0.0
Prioridad:	Media	**Código de requerimiento:**	R_3

Objetivo en contexto:	Modificar la contraseña de usuario.	
Actores:	Usuarios	
Entradas:	Los datos de usuario almacenados dentro de la base de datos.	
Salidas:	Contraseña modificada.	
Pre-condiciones:	Estar registrado en el sistema e iniciado sesión.	
Post-condiciones:	**Final de éxito:**	Contraseña modificada.
	Final de fallo:	Reintentar o comunicar con soporte de software.

Flujo básico de éxito

No.	**Actor**	**No**	**Sistema**
1	El usuario debe cumplir con el procedimiento escrito en IG_1.		
2	El usuario se dirige a su perfil para modificar su registro.		
3	Ingresa su nueva contraseña		
4	Activa la función para actualizar		
		5	Se almacena la nueva contraseña en la base de datos

Cuadro 11. Descripción de caso de uso validar contraseña

Id caso de uso:	IG-3		
Nombre:	Validar contraseña.		
Proyecto:	Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.		
Autor:	Santos Jeremías García Tzul	**Versión:**	1.0.0.0
Prioridad:	Media	**Código de requerimiento:**	R_2

Objetivo en contexto:	Valida el ingreso de las credenciales para poder acceder al módulo del sistema.	
Actores:	Usuario	
Entradas:	Descritos en IG-1	
Salidas:	Ingreso hacia el módulo administración.	
Pre-condiciones:	Estar registrado en el sistema.	
Post-condiciones:	**Final de éxito:**	Accede al sistema.
	Final de fallo:	Ingresa a la página de Login para intentar de nuevo.

Flujo básico de éxito

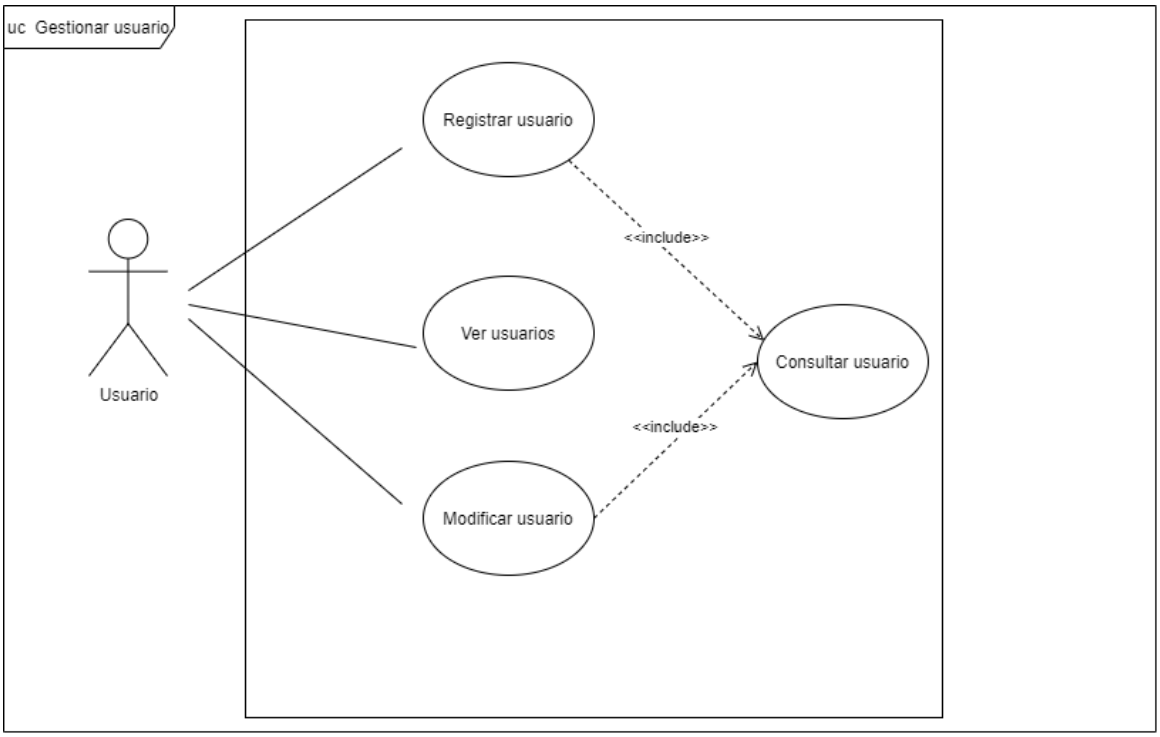
No.	**Actor**	**No**	**Sistema**
1	Usuario realiza el procedimiento de IG-1.		
		2	El sistema consulta a la base de datos, si los datos coinciden retorna un valor verdadero.
		3	Permite el ingreso al módulo.

Caminos alternativos	Si el sistema retorna un falso, usuario realiza el procedimiento de IG-1.
Caminos de excepción	Existe algún error el sistema deberá mostrar mensaje sobre el error.

6. Caso de uso gestionar usuario.

Dentro del caso de uso se presentan las funcionalidades que podrán realizar en la gestión de los usuarios dentro del sistema.

Ilustración 51. Diagrama de caso de uso para la gestión de usuario



Del cuadro 12 al 15 se describen cada uno de los casos de uso para la gestión de los usuarios.

Cuadro 12. Descripción de caso de uso buscar usuario

Id caso de uso:	CU-1		
Nombre:	Buscar usuario		
Proyecto:	Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.		
Autor:	Santos Jeremías García Tzul	**Versión:**	1.0.0.0
Prioridad:	Alta	**Código de requerimiento:**	R_4

Objetivo en contexto:	Buscar un usuario en el sistema.	
Actores:	Usuarios	
Entradas:	Id usuario.	
Salidas:	Retorna datos relacionado al usuario.	
Pre-condiciones:	Iniciado sesión al sistema.	
Post-condiciones:	**Final de éxito:**	Retornar un valor verdadero.
	Final de fallo:	Retorna un valor falso.

Flujo básico de éxito

No.	**Actor**	**No**	**Sistema**
1	Usuario ingresa el id del nuevo usuario.		
		2	El sistema encontró coincidencias retornara un valor verdura
		3	El sistema no encontró coincidencia retornara un valor falso.

Cuadro 13. Descripción de caso de uso registrar un usuario

Id caso de uso:	CU-2		
Nombre:	Registrar un usuario		
Proyecto:	Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.		
Autor:	Santos Jeremías García Tzul	**Versión:**	1.0.0.0
Prioridad:	Alta	**Código de requerimiento:**	R_4

Objetivo en contexto:	Registrar a un usuario dentro de la base de datos para poder ingresar al sistema y administrar el módulo de administrador.	
Actores:	Usuario	
Entradas:	Nombre, usuario, Contraseña y avatar.	
Salidas:	Usuario registrado	
Pre-condiciones:	Iniciado sesión al sistema.	
Post-condiciones:	**Final de éxito:**	Usuario registrado
	Final de fallo:	Comenzar de nuevo con el caso de uso.

Flujo básico de éxito

No.	**Actor**	**No**	**Sistema**
1	Usuario ingresa datos del nuevo usuario		
		2	El sistema ejecuta CU-1
		3	Si el resultado de CU-1 es verdadero se prosigue a guardar la información en la base de datos.

Caminos alternativos	Si el usuario ya existe no dejara registrar.
Caminos de excepción	Mostrar que el nombre de usuario ya está en uso.

Cuadro 14. Descripción de caso de uso ver usuarios registrado

Id caso de uso:	CU-3		
Nombre:	Ver usuarios registrados		
Proyecto:	Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.		
Autor:	Santos Jeremías García Tzul	**Versión:**	1.0.0.0
Prioridad:	Media	**Código de requerimiento:**	R_6
Objetivo en contexto:	Consultar los usuarios registrados.		
Actores:	Usuario		
Entradas:	Ninguno		
Salidas:	Todos los usuarios registrados		
Pre-condiciones:	Iniciado sesión al sistema.		
Post-condiciones:	**Final de éxito:**	Mostrar todos los usuarios.	
	Final de fallo:	Regresar al caso de uso CU-2	

Flujo básico de éxito			
No.	**Actor**	**No**	**Sistema**
1	El usuario ingresa al módulo administración.		
		2	El sistema retornara todos los datos de los usuarios.

Caminos alternativos	El usuario quiere hacer modificación a un usuario registrado deberá ir al caso de uso CU-4.
Caminos de excepción	Si se genera error mostrar al usuario el mensaje.

Cuadro 15. Descripción de caso de uso modificar usuario

Id caso de uso:	CU-4		
Nombre:	Modificar usuario		
Proyecto:	Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.		
Autor:	Santos Jeremías García Tzul	**Versión:**	1.0.0.0
Prioridad:	Medio	**Código de requerimiento:**	R_5

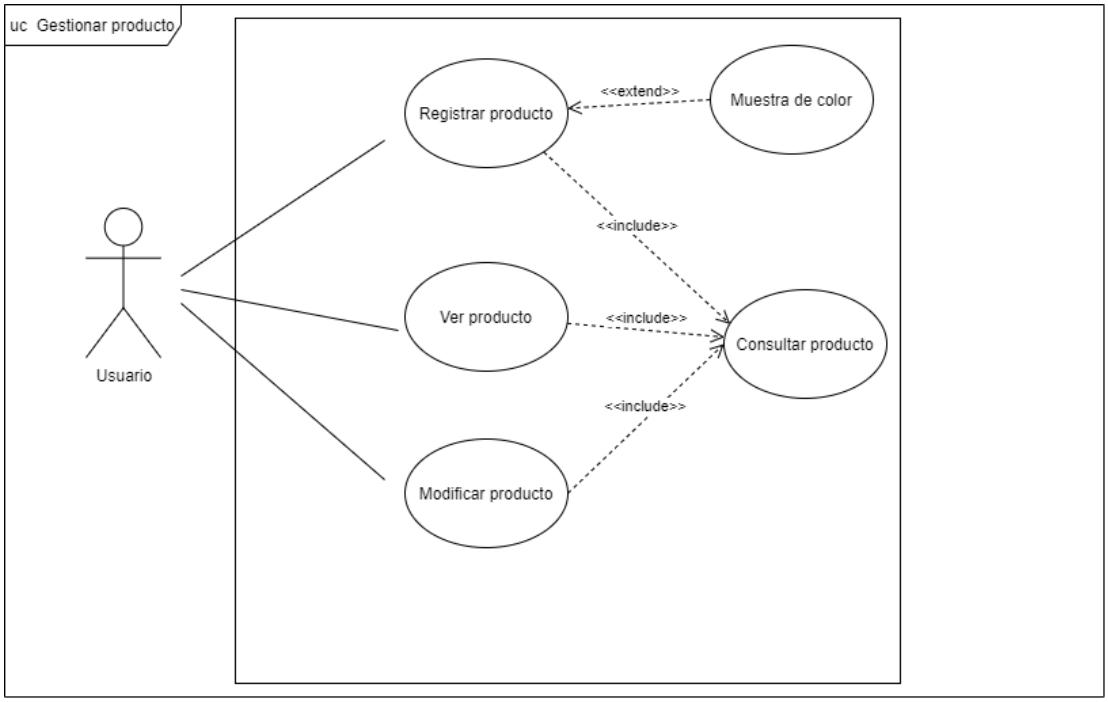
Objetivo en contexto:	Modificar un usuario registrado.		
Actores:	Usuarios		
Entradas:	Id usuarios, nombre, usuario, contraseña y avatar.		
Salidas:	Usuario actualizado		
Pre-condiciones:	Iniciado sesión.		
Post-condiciones:	**Final de éxito:**	Usuario actualizado.	
	Final de fallo:	Ir al caso de uso CU-3	

Flujo básico de éxito			
No.	**Actor**	**No**	**Sistema**
1	Usuario ingresa el id usuario.		
		2	Ejecuta CU-1
		3	Si el resultado de CU-1 es verdadero se prosigue a guardar la información en la base de datos.
		4	Se actualiza los datos del usuario.
Caminos alternativos	Se realizó correctamente la actualización se dirige a CU-3.		
Caminos de excepción	Si se genera error mostrar al usuario el mensaje.		

7. Caso de uso gestionar producto.

Dentro del caso de uso se presentan las funcionalidades que podrán realizar en la gestión de los productos dentro del sistema.

Ilustración 52. Diagrama de caso de uso para la gestión de productos



Del cuadro 16 al 20 se describen cada uno de los casos de uso para la gestión de los productos.

Cuadro 16. Descripción de caso de uso consultar producto

Id caso de uso:	GP-1		
Nombre:	Consultar producto.		
Proyecto:	Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.		
Autor:	Santos Jeremías García Tzul	**Versión:**	1.0.0.0
Prioridad:	Medio	**Código de requerimiento:**	R_7

Objetivo en contexto:	Buscar un producto a través de su id.		
Actores:	Usuario y clientes.		
Entradas:	Id de producto		
Salidas:	Todos los datos del producto consultado.		
Pre-condiciones:	Usuario: iniciado sesión y cliente entrado al sistema.		
Post-condiciones:	**Final de éxito:**	Mostrar los datos del producto.	
	Final de fallo:	Ir al caso de uso GP-3	

Flujo básico de éxito			
No.	**Actor**	**No**	**Sistema**
1	Usuario ingresa el id.		
		2	Sistema busca el producto con el id ingresado.
		3	Retornará el resultado de la consulta.

Caminos alternativos	Si se realizó correctamente la consulta retornará los datos.
Caminos de excepción	Sí se genera error mostrar al usuario el mensaje y en caso de que el producto no está registrado ir caso de uso GP-3.

Cuadro 17. Descripción de caso de uso muestra de color

Id caso de uso:	GP-2		
Nombre:	Toma de muestra de colores.		
Proyecto:	Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.		
Autor:	Santos Jeremías García Tzul	**Versión:**	1.0.0.0
Prioridad:	Alta	**Código de requerimiento:**	R_8

Objetivo en contexto:	De la imagen cargada por el usuario se le deberá tomar 5 muestra de colores y retornar los datos.	
Actores:	Usuario.	
Entradas:	Imagen base64.	
Salidas:	Un array de colores.	
Pre-condiciones:	Iniciado sesión al sistema.	
Post-condiciones:	**Final de éxito:**	Envía la muestra de colores.
	Final de fallo:	Regresar al caso de uso GP-3

Flujo básico de éxito

No.	**Actor**	**No**	**Sistema**
1	Usuario carga la imagen.		
		2	Sistema lo convierte a base64
		3	Realiza la operación para recorrer la imagen y retornar la muestra de colores.

Caminos alternativos	Sí se realizó correctamente la toma de colore retornar datos al caso de uso GP-3.
Caminos de excepción	Si se genera error mostrar al usuario el mensaje.

Cuadro 18. Descripción de caso de uso ingresar producto

Id caso de uso:	GP-3		
Nombre:	Registrar producto.		
Proyecto:	Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.		
Autor:	Santos Jeremías García Tzul	**Versión:**	1.0.0.0
Prioridad:	Alta	**Código de requerimiento:**	R_7

Objetivo en contexto:	Registrar un producto de la tienda.	
Actores:	Usuario.	
Entradas:	Categoría, nombre, estilo, típico, talla, cantidad, precio, existencia, caso de uso GP-2.	
Salidas:	El registro del producto en la base de datos.	
Pre-condiciones:	Iniciado sesión en el sistema.	
Post-condiciones:	**Final de éxito:**	Producto registrado.
	Final de fallo:	Reintentar de nuevo.

Flujo básico de éxito

No.	**Actor**	**No**	**Sistema**
1	Usuario ingresa los datos en los campos correspondiente.		
		2	Sistema ejecuta GP-2
		3	Sistema realiza la inserción correspondiente.

Caminos alternativos	Se realizó correctamente el registro regresar caso de uso GP-3.
Caminos de excepción	Sí se genera error mostrar al usuario el mensaje.

Cuadro 19. Descripción de caso de uso ver productos

Id caso de uso:	GP-4		
Nombre:	Ver producto del sistema.		
Proyecto:	Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.		
Autor:	Santos Jeremías García Tzul	**Versión:**	1.0.0.0
Prioridad:	Medio	**Código de requerimiento:**	R_9

Objetivo en contexto:	Mostrar todos los registros de los productos almacenados en la base de datos.	
Actores:	Usuario y cliente.	
Entradas:	Ninguna.	
Salidas:	Todos los registros de los productos almacenados en la base de datos.	
Pre-condiciones:	Usuario iniciado sesión y cliente ingresado al sistema.	
Post-condiciones:	**Final de éxito:**	Mostrar todos los productos.
	Final de fallo:	Ir al caso de uso GP-3

Flujo básico de éxito

No.	**Actor**	**No**	**Sistema**
		1	Sistema realiza la consulta de los registros almacenados en la base de datos.

Caminos alternativos	Realizado la consulta: 1. El usuario quiere hacer modificación un producto registrado deberá ir al caso de uso GP-5. 2. El cliente quiere ver referencia para su traje ir al caso de uso GA-3
Caminos de excepción	Si se genera error mostrar al usuario el mensaje.

Caso de uso modificar producto.

Cuadro 20. Descripción de caso de uso modificar producto registrado

Id caso de uso:	GP-5		
Nombre:	Modificar producto registrado.		
Proyecto:	Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.		
Autor:	Santos Jeremías García Tzul	**Versión:**	1.0.0.0
Prioridad:	Medio	**Código de requerimiento:**	R_10

Objetivo en contexto:	Modificar los datos de un producto registrado.	
Actores:	Usuario.	
Entradas:	Id del producto.	
Salidas:	Registro de producto actualizado.	
Pre-condiciones:	Iniciado sesión en el sistema.	
Post-condiciones:	**Final de éxito:**	Producto actualizado
	Final de fallo:	Repetir caso de uso.

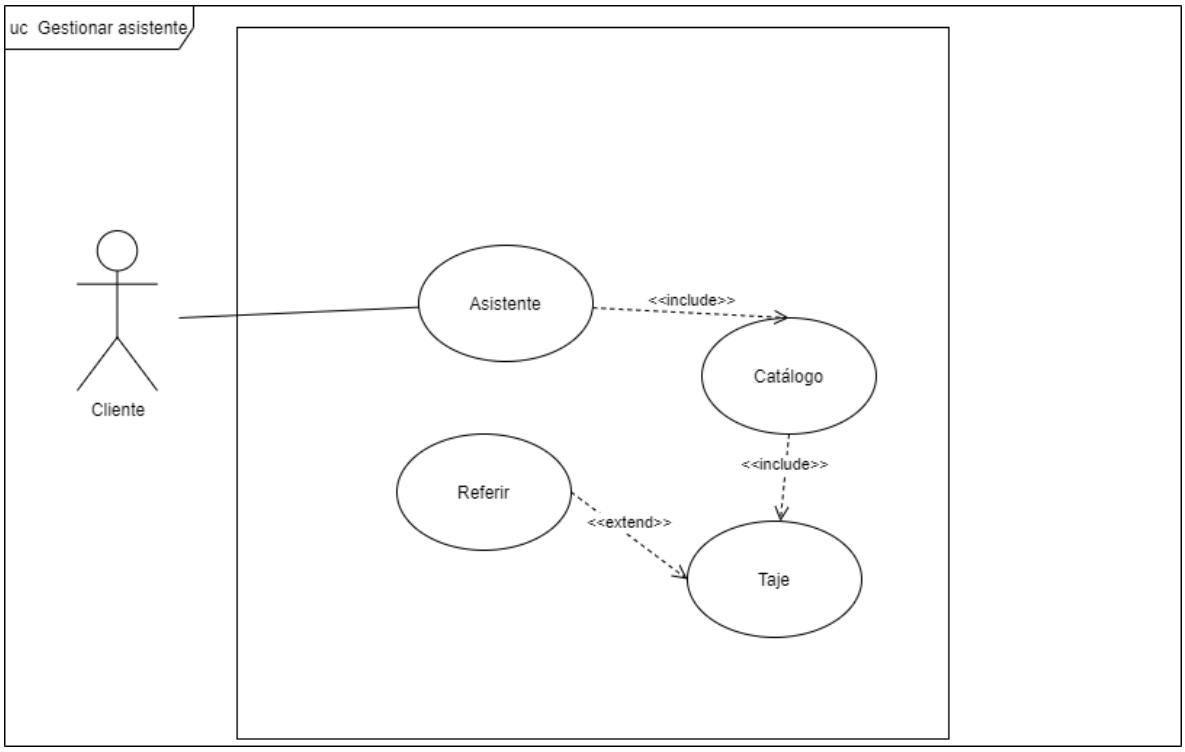
Flujo básico de éxito			
No.	**Actor**	**No**	**Sistema**
		1	El sistema ejecuta GP-1
2	Usuario ingresa el id del producto.		
		3	Realiza la actualización de los datos.

Caminos alternativos	Realizado la modificación se dirige al caso de uso GP-4.
Caminos de excepción	Sí se genera error mostrar al usuario el mensaje.

8. Caso de uso gestión de asistente.

Dentro del caso de uso se presentan las funcionalidades que podrán realizar en la gestión del asistente dentro del sistema.

Ilustración 53. Diagrama de caso de uso para la gestión del asistente



Del cuadro 21 al 24 se describen cada uno de los casos de uso para la gestión de del asistente.

Cuadro 21. Descripción de caso de uso asistente de búsqueda de producto

Id caso de uso:	GA-1		
Nombre:	Asistente de búsqueda de producto.		
Proyecto:	Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.		
Autor:	Santos Jeremías García Tzul	**Versión:**	1.0.0.0
Prioridad:	Alta	**Código de requerimiento:**	R_12

Objetivo en contexto:	Mostrar las categorías que se tiene y espera que el cliente seleccione una categoría en especial.	
Actores:	Cliente	
Entradas:	Categoría que necesita el cliente ver.	
Salidas:	Nombre de la categoría que necesita ver.	
Pre-condiciones:	Ingresar al sistema en modulo cliente.	
Post-condiciones:	**Final de éxito:**	Enviar la categoría al caso de uso GA-2.
	Final de fallo:	Reintentar caso de uso GA-1

Flujo básico de éxito

No.	**Actor**	**No**	**Sistema**
1	Ingresa en el módulo asistente.		
2	Selecciona la categoría que necesita ver.		
		3	Envía la categoría al caso de uso GA-2

Cuadro 22. Descripción de caso de uso catálogo

Id caso de uso:	GA-2		
Nombre:	Catálogo de producto según categoría seleccionada.		
Proyecto:	Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.		
Autor:	Santos Jeremías García Tzul	**Versión:**	1.0.0.0
Prioridad:	Media	**Código de requerimiento:**	R_13

Objetivo en contexto:	Mostrar los productos registrados dentro del sistema por categoría para que el cliente pueda seleccionar una.	
Actores:	Cliente.	
Entradas:	Caso de uso GA-1	
Salidas:	Mostrar todos los productos con un formato especial.	
Pre-condiciones:	Ingresar al sistema.	
Post-condiciones:	**Final de éxito:**	Muestra los resultados de la búsqueda de GA-1.
	Final de fallo:	Repetir caso de uso GA-1

Flujo básico de éxito

No.	**Actor**	**No**	**Sistema**
		1	El sistema toma los datos de caso de uso GA-1
		2	El sistema da formato a los datos a mostrar.

Caminos alternativos	Muestra los datos en pantalla. El usuario pude seleccionar la opción de traje para referir los que coinciden caso de uso GA-4.
Caminos de excepción	Sí se genera error mostrar al usuario el mensaje.

Cuadro 23. Descripción de caso de uso referir

Id caso de uso:	GA-3		
Nombre:	Referir productos que tengan colores similares.		
Proyecto:	Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.		
Autor:	Santos Jeremías García Tzul	**Versión:**	1.0.0.0
Prioridad:	Alta	**Código de requerimiento:**	R_14

Objetivo en contexto:	Referir al cliente los productos que tengan coincidencia en color.	
Actores:	Cliente	
Entradas:	Colores de la imagen del producto seleccionadas por el cliente.	
Salidas:	Todos los productos que tenga coincidencia en los colores.	
Pre-condiciones:	Ingresar al sistema.	
Post-condiciones:	**Final de éxito:**	Mostrar todos los productos que tenga coincidencia.
	Final de fallo:	Ir al caso de uso GA-2.

Flujo básico de éxito

No.	**Actor**	**No**	**Sistema**
1	Usuario selecciona el producto que quiere combinar con otros productos.		
		2	Sistema realizará una consulta de las coincidencias que encuentre.

Caminos alternativos	Muestra los datos en pantalla para el caso de uso GA-4.
Caminos de excepción	Sí se genera error mostrar al usuario el mensaje.

Cuadro 24. Descripción de caso de uso traje

Id caso de uso:	GA-4			
Nombre:	Traje.			
Proyecto:	Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL.			
Autor:	Santos Jeremías García Tzul	**Versión:**	1.0.0.0	
Prioridad:	Alta	**Código de requerimiento:**	R_15	
Objetivo en contexto:	Mostrar los productos coincidentes segmentando por categoría con una vista previa de cómo queda.			
Actores:	Cliente.			
Entradas:	Caso de uso GA-3.			
Salidas:	Muestra los datos por categoría.			
Pre-condiciones:	Ingresar al sistema.			
Post-condiciones:	**Final de éxito:**	Muestra los productos.		
	Final de fallo:	Ir al caso de estudio GA-2		
Flujo básico de éxito				
No.	**Actor**	**No**	**Sistema**	
1	Cliente según esta seleccionado en caso de uso GA-2.			
		2	Sistema ejecuta caso de uso GA-3.	
Caminos alternativos	Muestra los datos en pantalla.			
Caminos de excepción	Si se genera error mostrar al usuario el mensaje.			

D. CONSTRUCCIÓN DEL PROTOTIPO

Se presenta la estructura que tendrá el sistema web basados en el diseño empleado con anterioridad. Presenta las características, diseños, funcionalidades y estructura de la base de datos.

1. Alcance y restricciones.

Se definió las limitaciones del sistema y como las funcionalidades que presenta, en las restricciones se define acorde al tipo de sistema que se desarrolló.

a. Alcance del sistema

El sistema que se desarrolló para la tienda de ropa típica se encarga de gestionar los productos que ofrecen, brindando un asistente para los clientes donde cuenta con un catálogo y un sistema de referencias en base a los colores que tenga el producto seleccionado. Este sistema web se emplea en una red local, las funcionalidades del sistema serán los siguientes:

1. Módulo de administración.
2. Módulo de cliente.

b. Restricciones del sistema

Basado en los casos de uso descritos con anterioridad, se presentan las siguientes restricciones para el sistema.

- El sistema no dejará ingresar a las rutas donde se gestionen todo lo relevante a los productos para los clientes.
- El sistema solo podrá ser empleado en una red de are local.
- Las funcionalidades serán aceptables ya que es un modelo de prototipos.

2. Características del sistema.

Teniendo en cuenta el alcance y los requerimientos presentados podemos establecer la estructura que contará el sistema basando en los siguientes componentes para su desarrollo.

- **Tipo de sistema:** sistema web basado en una red local.
- **Restricciones:** el ingreso autorizado hacia el servidor donde almacenará la información y modificaciones que requiera el sistema deberán consultar con el desarrollador.
- **Lenguajes de programación:** HTML, JavaScript, PHP, Framework Laravel y Framework Bootstrap.
- **Sistema operativo de servidor:** Windows 10 empleando XAMPP.
- **Servidor de aplicación:** Apache integrada de XAMPP.
- **Gestor de base de datos:** MySQL versión 10.

a. Roles y funcionalidades

Para los roles y como funcionalidad del sistema se emplean dos, uno para el usuario y el otro para el cliente.

- **Rol del usuario:** contará con el rol de administración de los productos teniendo a su disposición realizar inserción, consulta y actualización.
- **Funcionalidad del usuario:** podrá ingresar al módulo de administración que está conformada por registro de productos, inventario, modificar producto y el módulo de usuario que de igual forma podrá registrar, actualizar un usuario.
- **Rol de cliente:** únicamente podrá realizar consultas.
- **Funcionalidad del cliente:** podrá ingresar al módulo de cliente, conformada por asistente, catálogo y referencia.

3. Dependencias del sistema web.

a. Software

Dentro del servidor debe tener instalado XAMPP para poder ejecutar el servidor apache y el gestor de base de datos MySQL.

Para el cliente debe tener instalado el navegador Microsoft Edge, Google Chrome, en caso de ser un dispositivo móvil con Google Chrome funciona sin ningún problema.

b. Hardware

Deberá contar una estructura de red local y debe comprender los requisitos de servidor y del equipo para el cliente según los requerimientos de hardware.

4. Estructura del sistema web.

Se estructuró en dos áreas siendo estos la de FRONTEND conocido como interfaz de usuario y el BACKEND que es toda la lógica que está detrás del sistema, donde se gestionará cada petición del usuario y del cliente que requieran el uso del sistema.

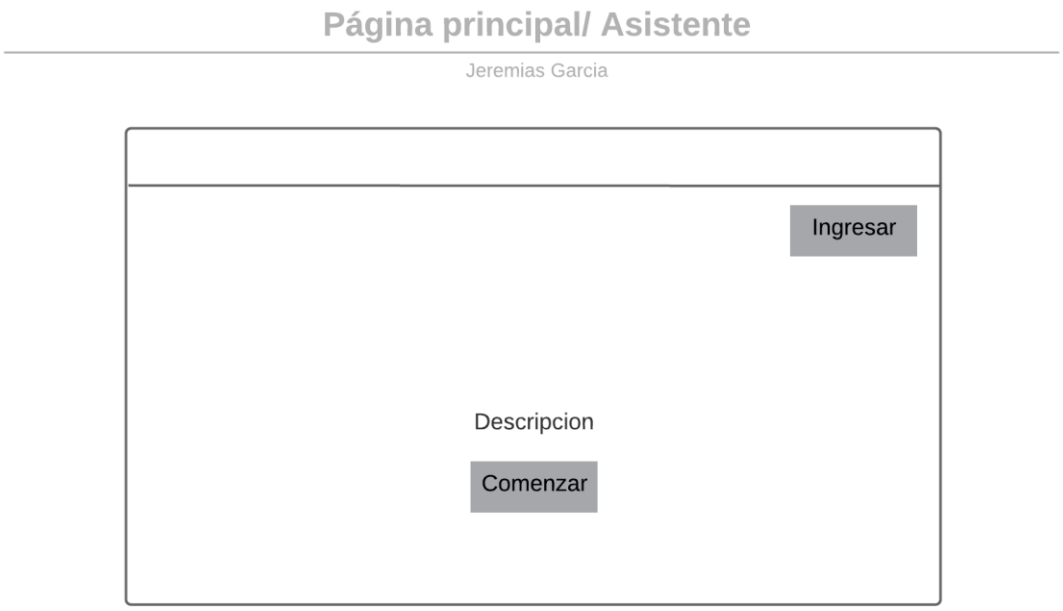
a. FRONTEND

Para el interfaz de usuario se estructuro en dos módulos, uno para el usuario y otro para el cliente ya que estos tendrán funcionalidades distintas como se presentó en el caso de uso.

Para el interfaz de usuario del área de usuario contará con las siguientes vistas:

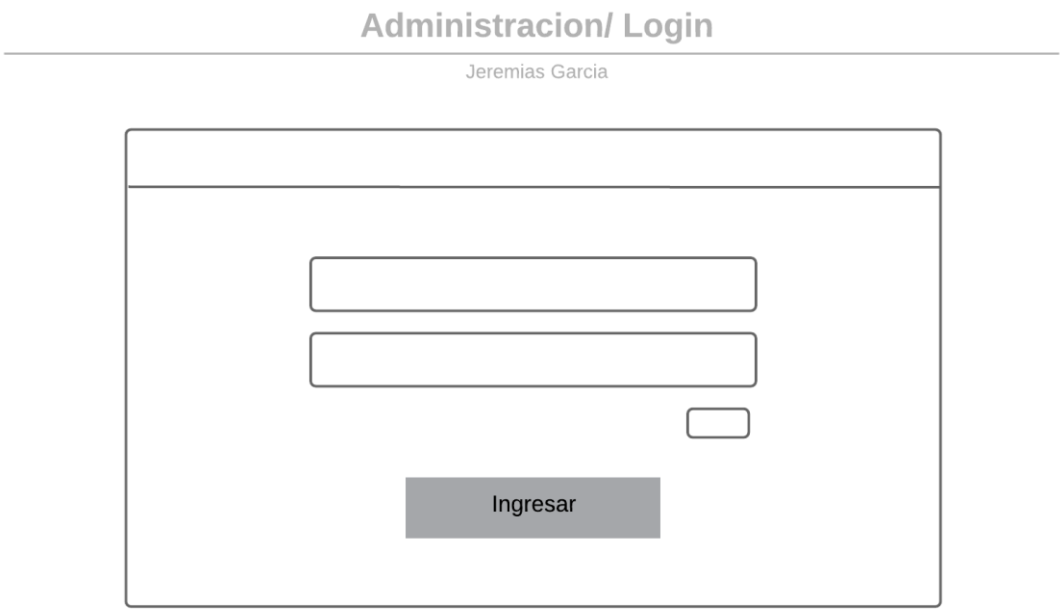
Home: donde encontrarán un botón de ingreso al módulo de administración.

Ilustración 54. Interfaz de usuario home



Login: donde ingresarán los credenciales para poder ingresar al módulo administración.

Ilustración 55. Interfaz de usuario login

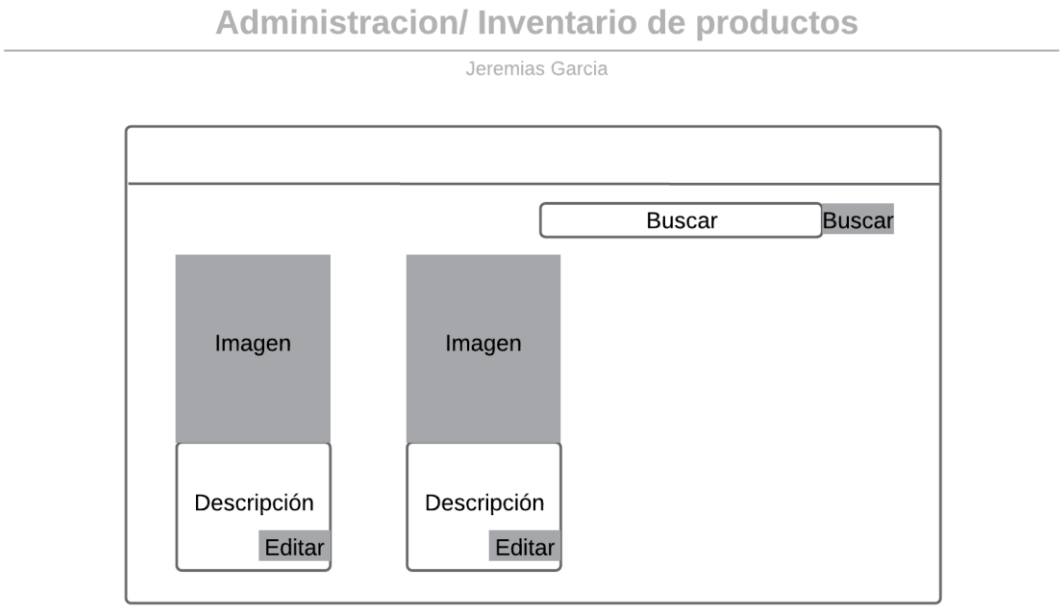


Administración-registro: donde podrán registrar los productos después de autenticar los credenciales.

Ilustración 56. Interfaz de usuario registrar producto

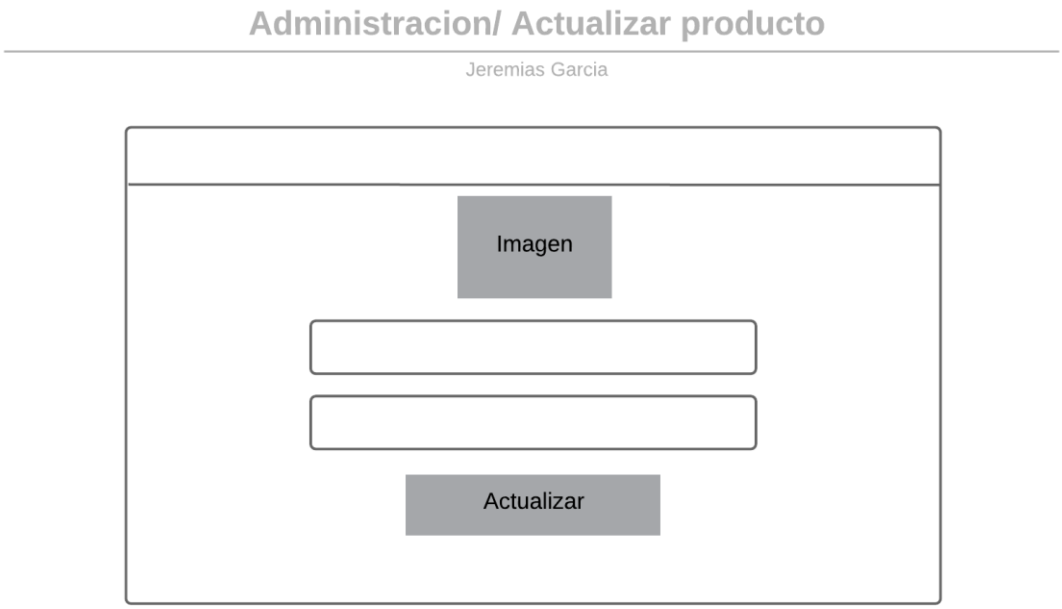
Administración-inventario: muestra los productos registrados, podrá realizar consulta específica y los productos que se presentan podrán ser modificado.

Ilustración 57. Interfaz de usuario inventario



Administración-actualizar: donde pueden modificar los datos de un producto.

Ilustración 58. Interfaz de usuario actualizar producto

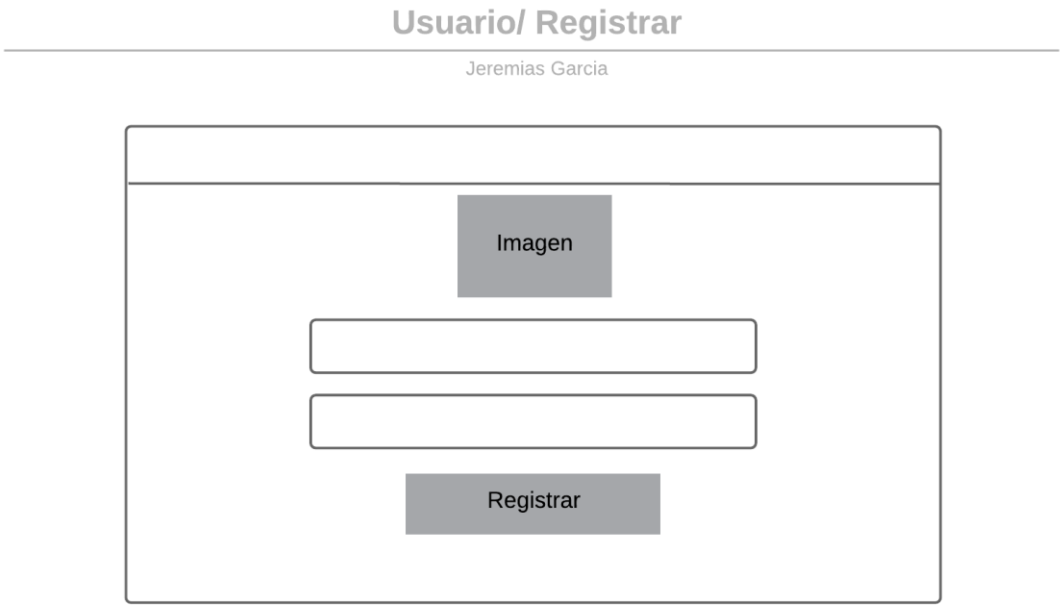


Usuario-ver: mostrará los usuarios registrados en el sistema.

Ilustración 59. Interfaz de usuario ver usuarios

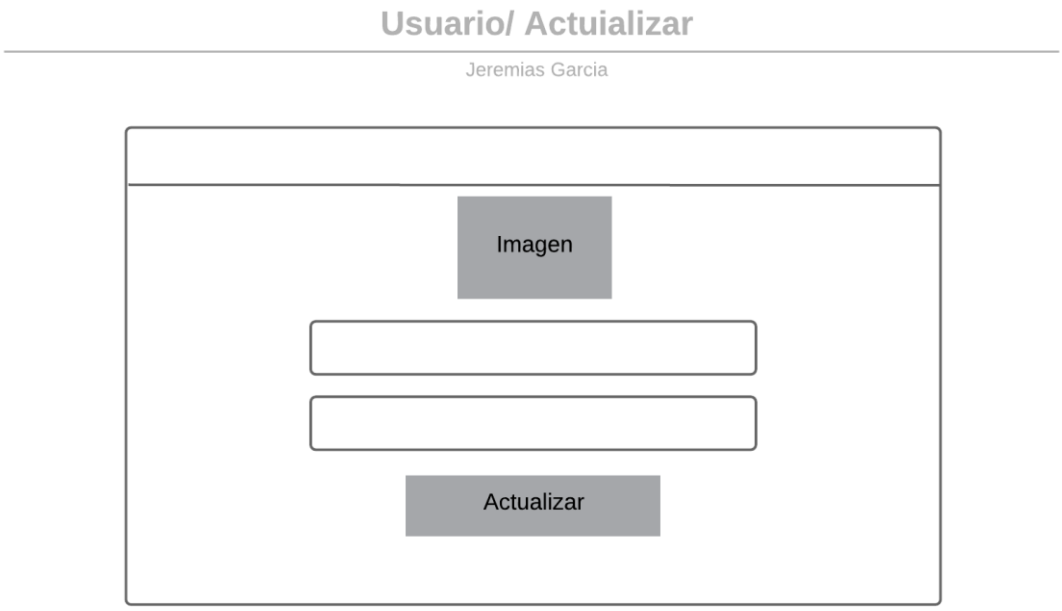
Usuario-registrar: donde podrán registrar a los usuarios del sistema.

Ilustración 60. Interfaz de usuario registrar usuario



Usuario-actualizar: podrán actualizar los datos de los usuarios.

Ilustración 61. Interfaz de usuario actualizar usuario

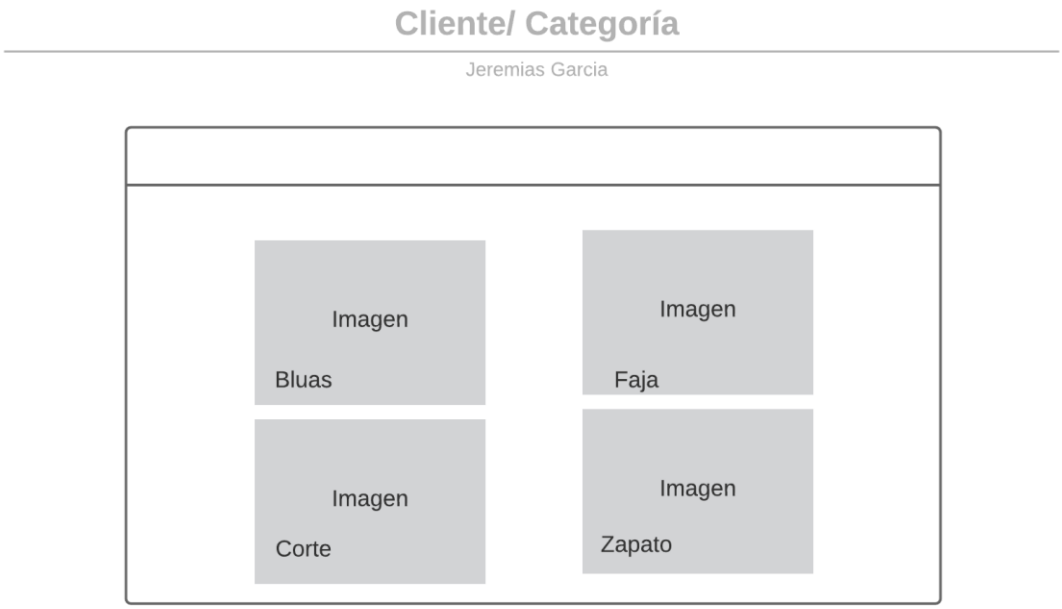


Para las vistas que tendrá el cliente son los siguientes:

Home: para el cliente únicamente puede accionar la opción comenzar para que el asistente ayude al cliente en la búsqueda del producto, puede consultar Ilustración 54.

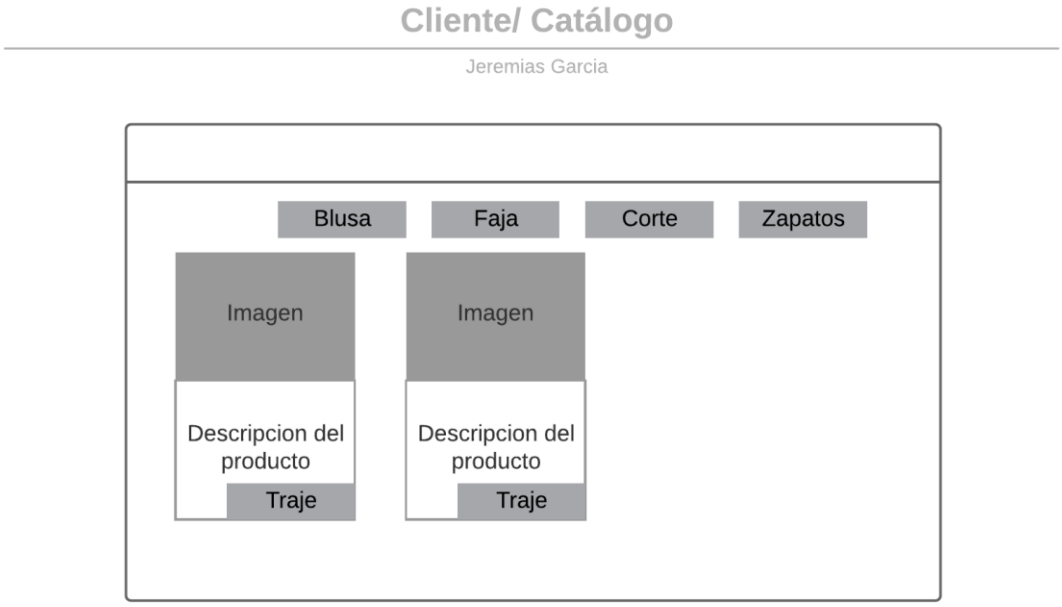
Menú de categorías: muestra las categorías de productos que se tiene en la tienda y podrá accionar sobre una de estas.

Ilustración 62. Interfaz de usuario menú de categorías



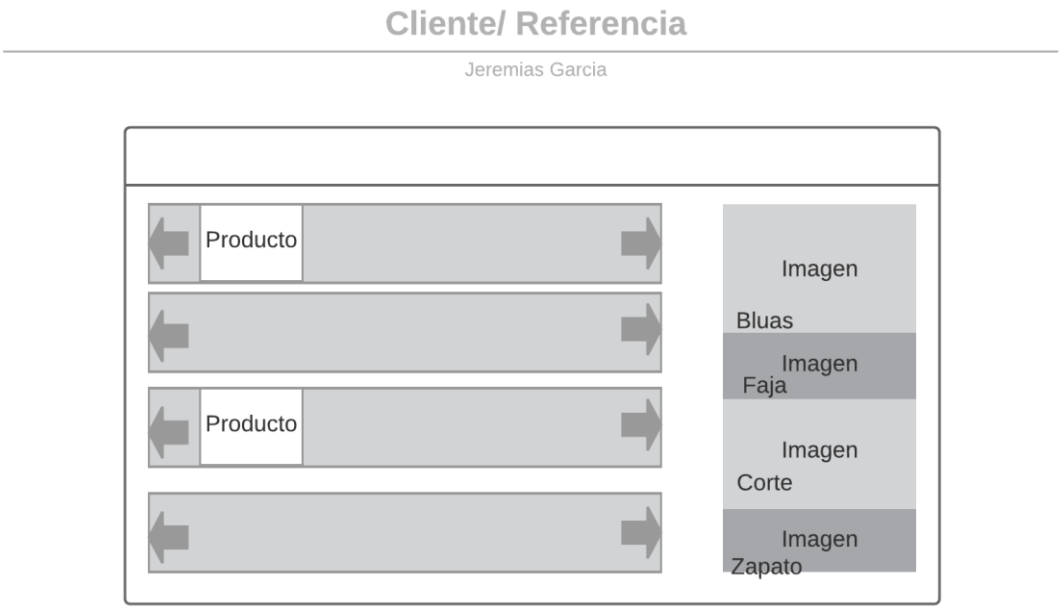
Catálogo: donde se muestra todos los productos segmentados por categorías que el usuario seleccionó y además podrá cambiar de categoría si lo llegase a necesitar. Dentro de los productos mostrados podrá acceder a completar su traje.

Ilustración 63. Interfaz de usuario catálogo



Referencia: Donde podrá combinar los cuatro productos.

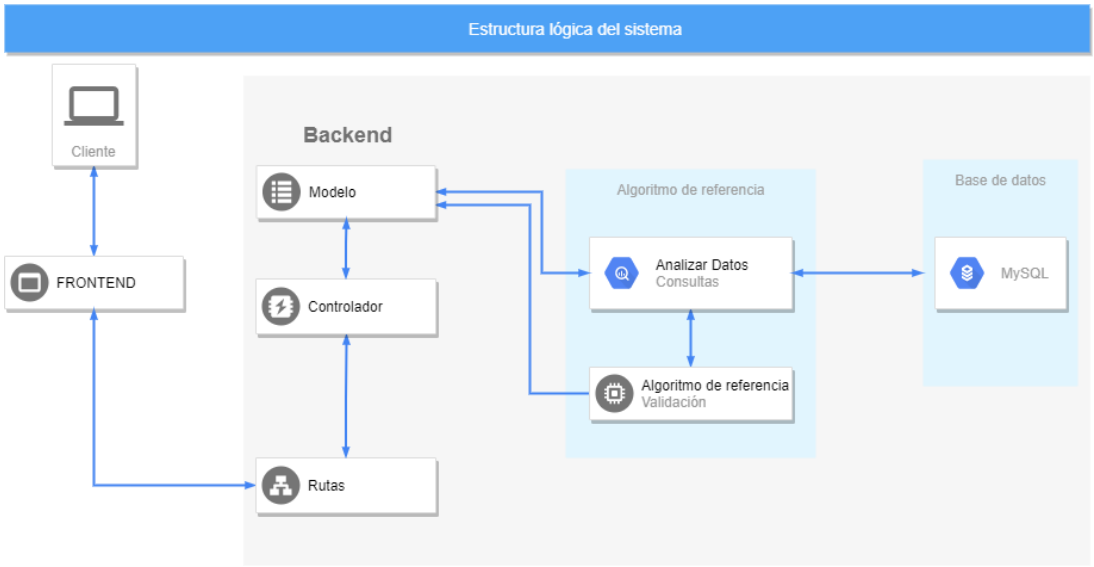
Ilustración 64. Interfaz de usuario referencia



b. BACKEND

La estructura lógica que se implementó desde la petición del usuario y el cliente hacia el servidor, las peticiones realizadas por el usuario es enviada hacia las rutas creadas e interpreta la acción que es requerida hacer, de las rutas se envían a los controladores quienes se encargan de administrar las consultas a la base de datos mediante los modelos que se empleó y que estos mismo modelos retorna los datos hacia los controladores y se encarga de enviar las información a la ruta donde fue requerida. En la siguiente ilustración veremos la estructura del sistema.

Ilustración 65. Estructura lógica del sistema

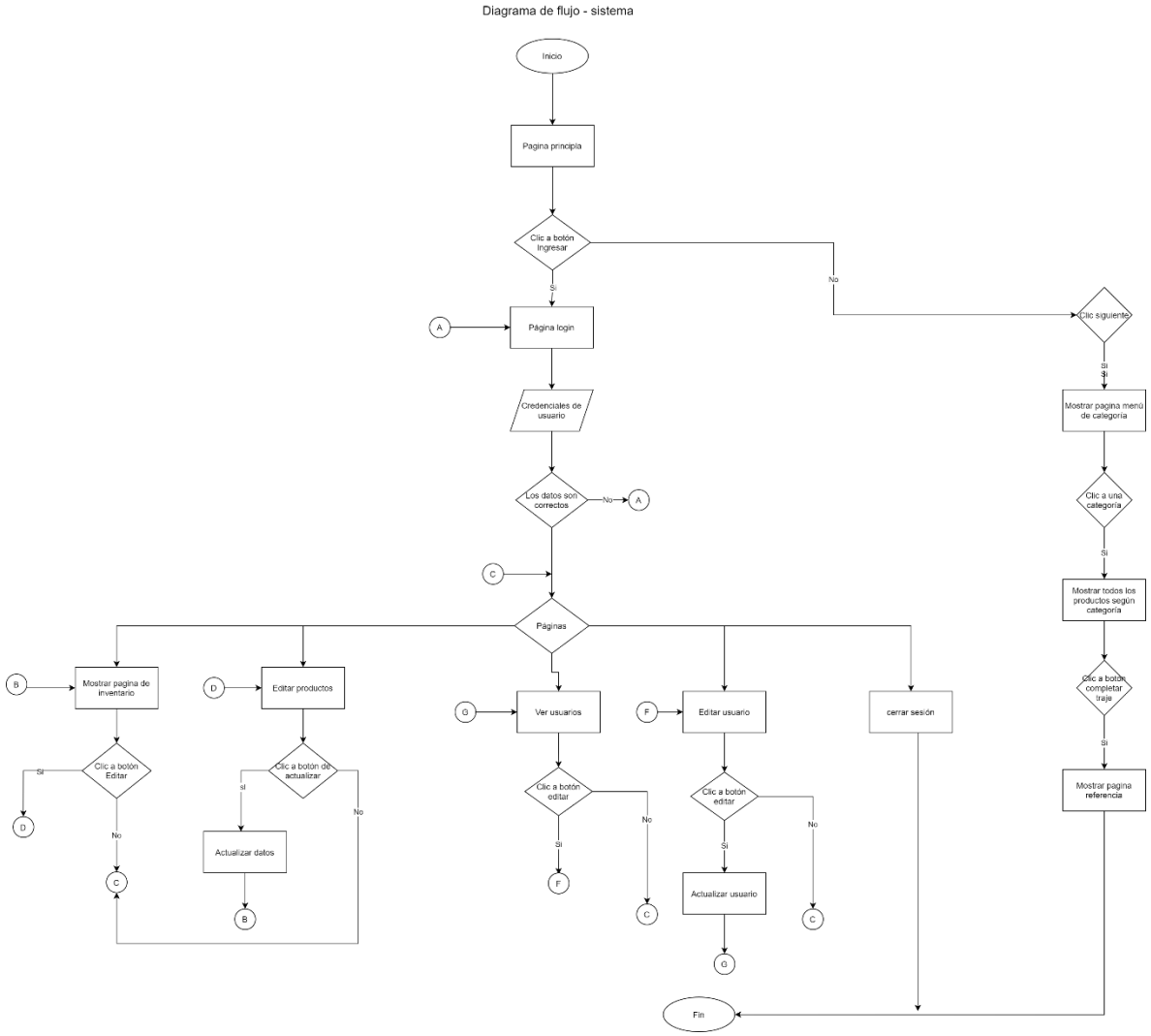


1) Diagrama de flujo

Se presenta el flujo que tendrá el sistema web, sábados en cada una de las funcionalidades mencionadas con anterioridad.

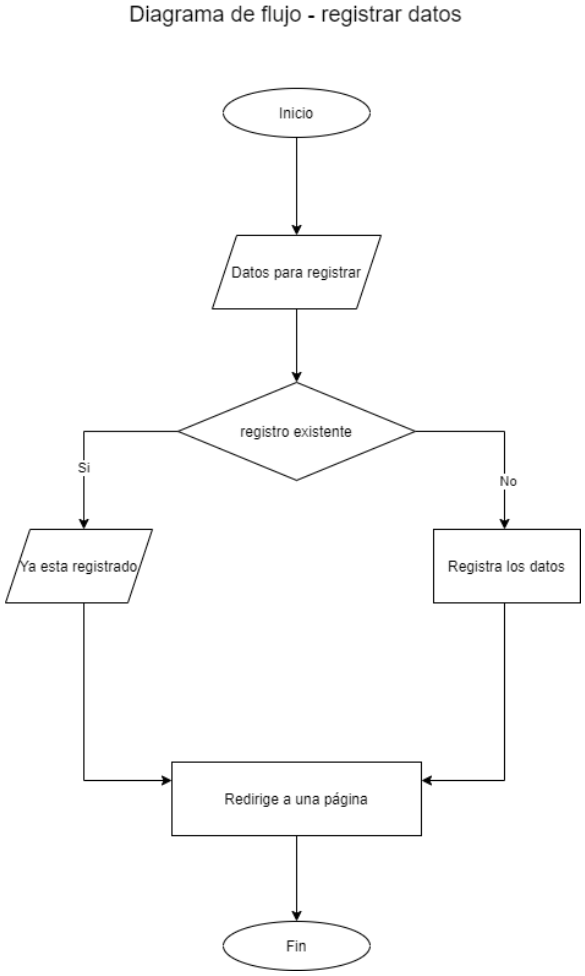
a) Sistema

Ilustración 66. Algoritmo del sistema



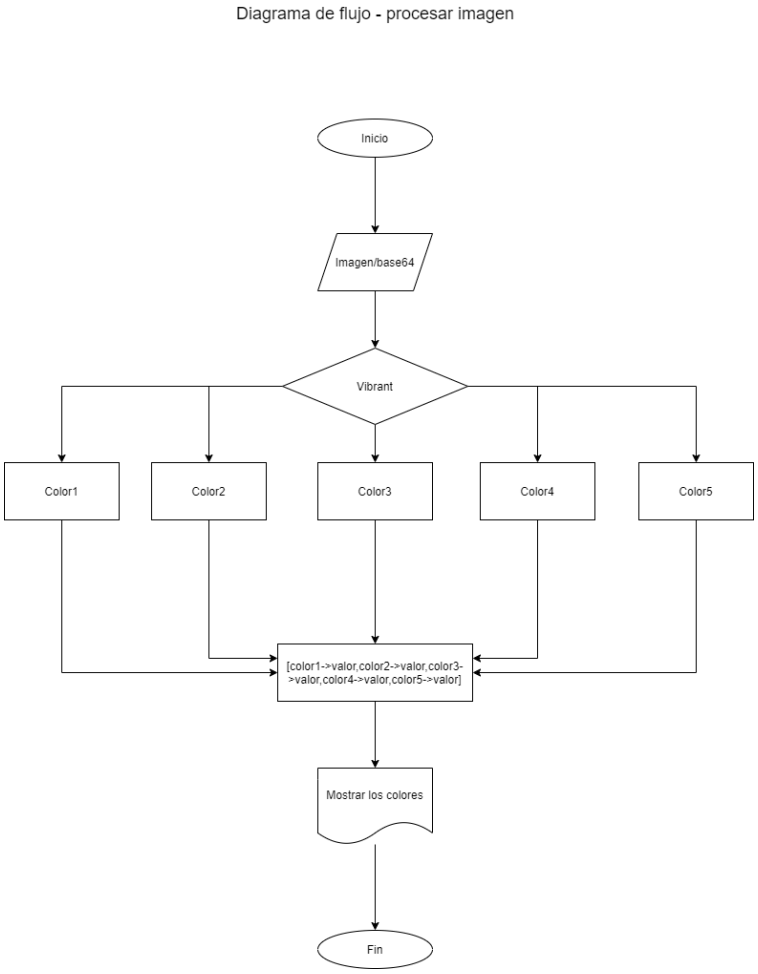
b) Registrar producto

Ilustración 67. Algoritmo para registrar datos



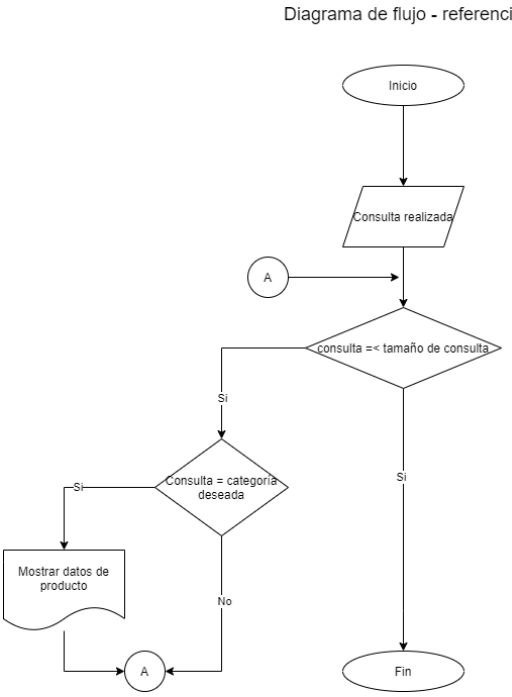
c) Muestra de colores

Ilustración 68. Algoritmo para tomar muestra de color de imagen



d) Referencia

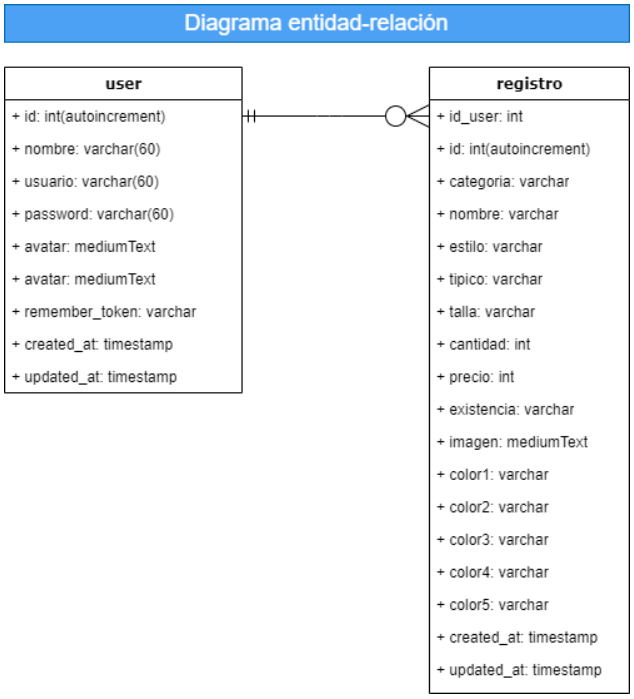
Ilustración 69. Diagrama de flujo de sistema de referencia



2) Estructura de la base de datos

Se almacenará los datos en dos tablas distintas una de estas será para el registro de los usuarios dentro de la base de datos y el otro para el registro de los productos que tiene la tienda.

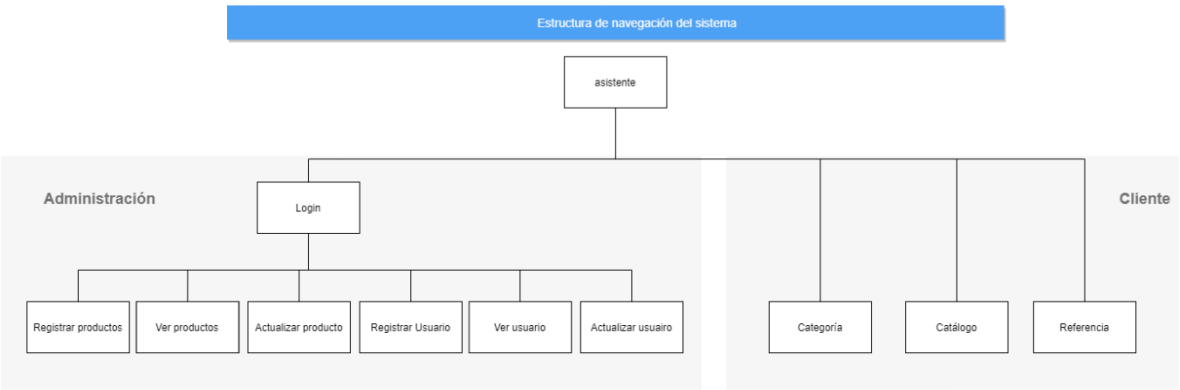
Ilustración 70. Diagrama de base de datos



3) Rutas de navegación del sistema

Se presenta las rutas que tiene el sistema, de esta forma comprenderemos el comportamiento de éste al momento de ingresar dentro de una de las páginas.

Ilustración 71. Ruta de navegación del sistema



5. Herramientas de desarrollo.

Las herramientas seleccionadas son los que ayudaron a desarrollar el proyecto como lo es Visual estudio Code, Composer y XAMPP.

a. Servidor

Para el servidor se optó por la herramienta XAMPP que brinda un servicio que simula un servidor en una computadora común, que ayudó a alojar el proyecto de manera que se puede acceder al asistente desde la misma red al cual está el servidor.

Se debe instalar Composer para poder instalar las librerías de PHP, de esta manera se podrá crear el proyecto y gestionar el mismo.

b. Interfaz

Para el diseño del interfaz se seleccionó la hoja de estilos CSS, el Framework de Bootstrap V5 que ayudó al sistema que fuese RESPONSIVE y Cropper que ayudó en el are de recorte de imagen, cada uno de estos se pensó en base a los diseños creados en el modelo de diseño,

c. Lógica

Para la lógica del sistema se seleccionó los lenguajes JavaScript del lado del cliente y PHP del lado del servidor.

- **JavaScript**: ayudó en la toma de muestra de colores con la librería Vibran, Crooper para recortar la imagen al área necesaria y Jquery para controlar los DOMS del proyecto.
- **PHP**: Se seleccionó el Framework de Laravel que administra las rutas, las instrucciones necesarias para la inserción, consulta y modificación de los registros.

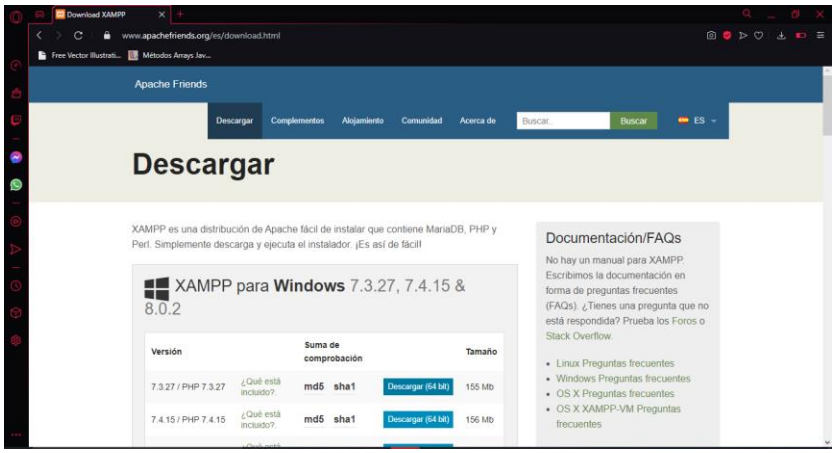
E. DESARROLLO DEL PROTOTIPO

1. Instalación de herramientas de trabajo

a. XAMPP

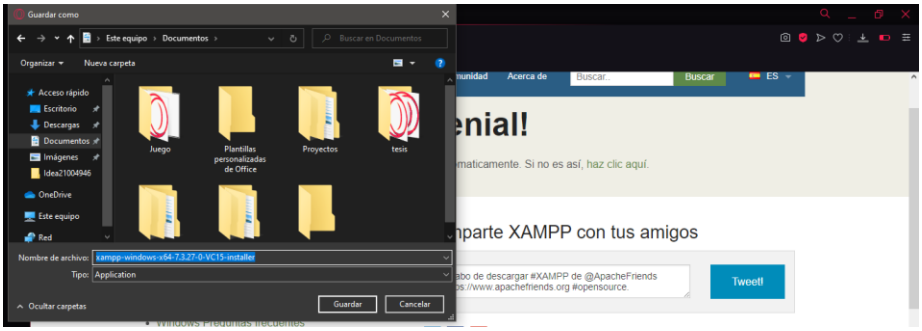
Para poder realizar la instalación es necesario descargar desde su página web oficial mediante este link https://www.apachefriends.org/download.html.

Ilustración 72. Página oficial para descargar XAMPP



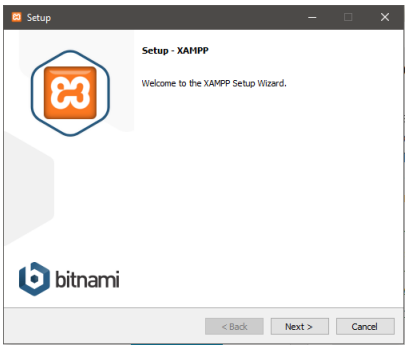
Es necesario descargar una versión mayor o igual a 7.3 de PHP en su versión de 64 bits para poder trabajar con los recursos del proyecto.

Ilustración 73. Guardar instalador



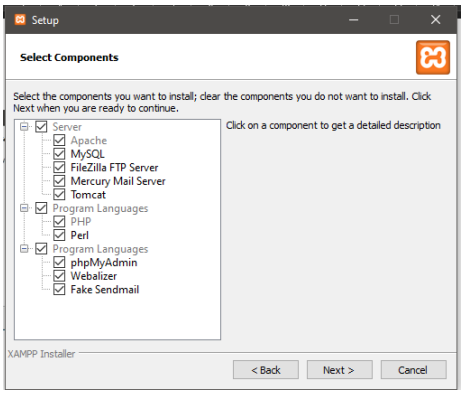
Ejecutar el instalador, nos mostrará un asistente de instalación donde le daremos Next.

Ilustración 74. Asistente de instalación de XAMPP



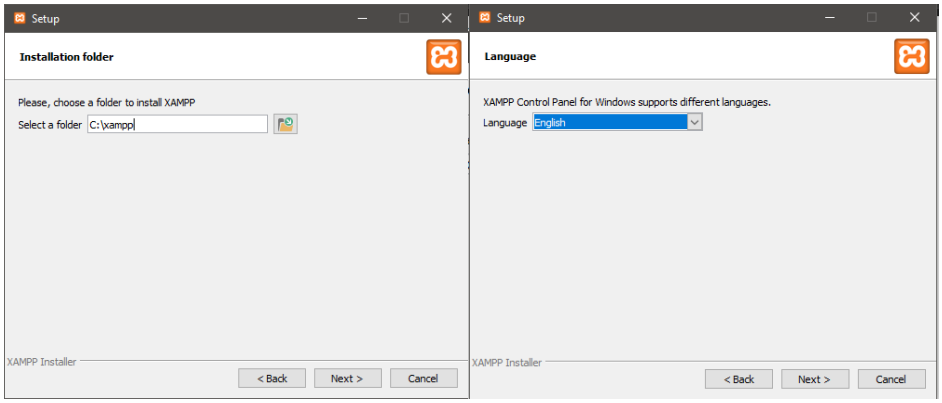
Revisamos que todos los componentes básicos están seleccionados que son Apache, MySQL, PHP y phpMyAdmin; después de revisar que los componentes están seleccionados, daremos clic a botón de Next.

Ilustración 75. Componentes de XAMPP



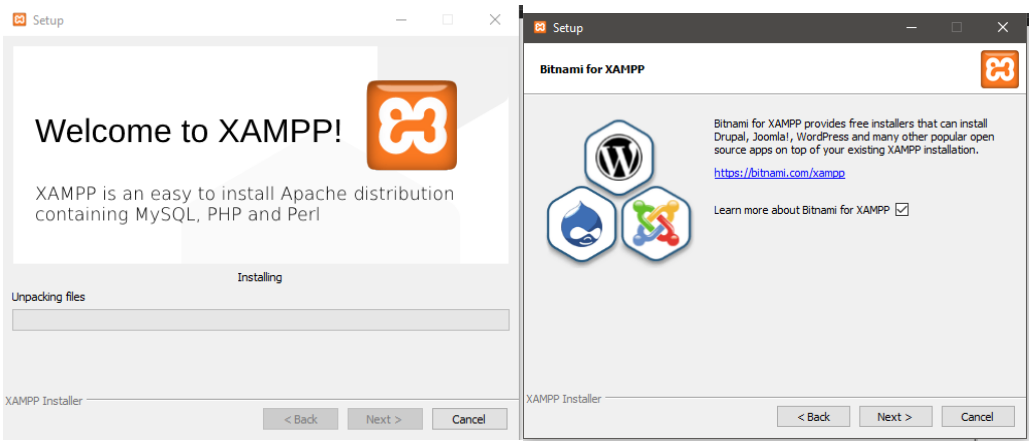
Revisamos que la ruta de instalación sea el correcto, luego daremos clic en el botón Next y revisamos que el idioma este en inglés.

Ilustración 76. Ruta de instalación y lenguaje



Esperamos a que se complete la instalación, para luego proseguir a dar Next

Ilustración 77. Proceso de instalación de XAMPP

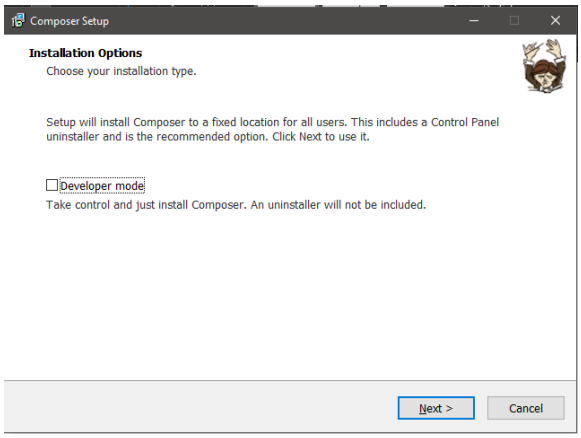


b. Composer

Descargar el ejecutable desde la página oficial, en la sección de Windows https://getcomposer.org/doc/00-intro.md.

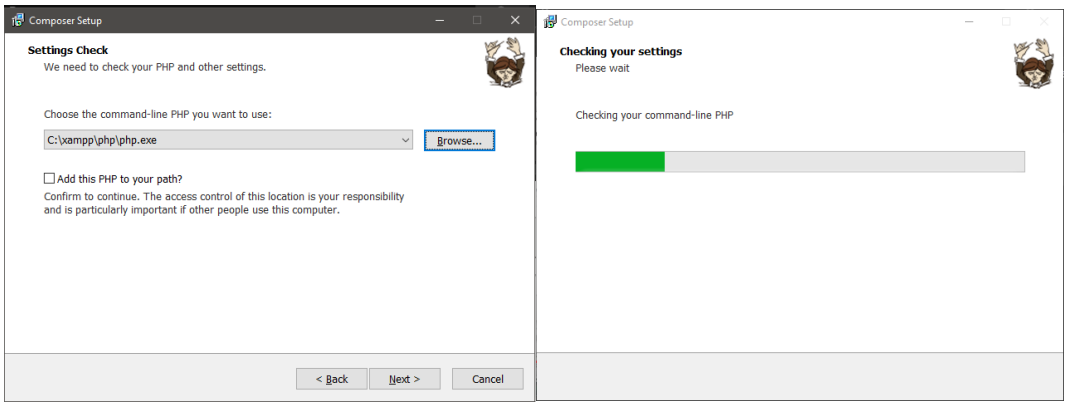
Seguir los pasos que muestra el asistente de instalación, presionamos sobre Next.

Ilustración 78. Asistente de instalación de Composer



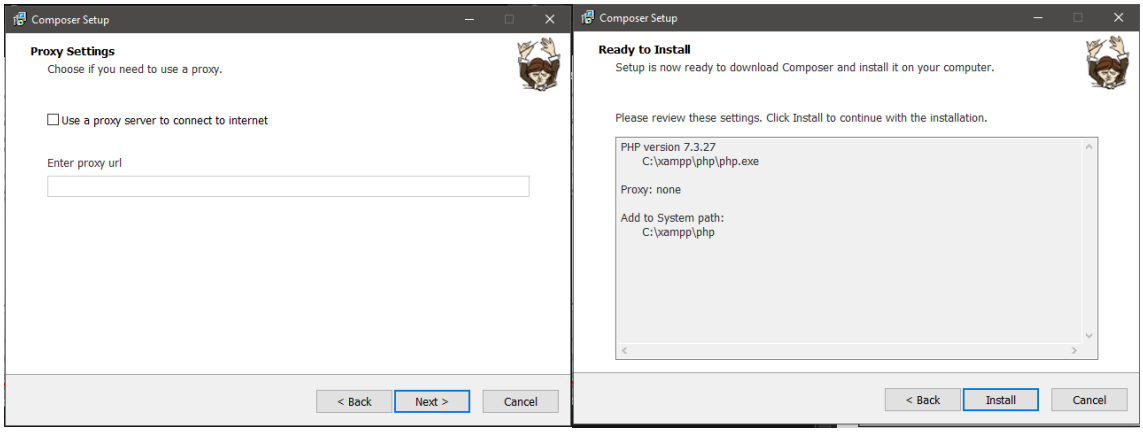
Verificamos la ruta de instalación que sea C:\xampp\php\PHP.exe ya que nuestro servidor será administrado por XAMPP y presionamos Next para que Composer verifique los componentes necesarios.

Ilustración 79. Verificación de ruta y proceso de validación de Composer



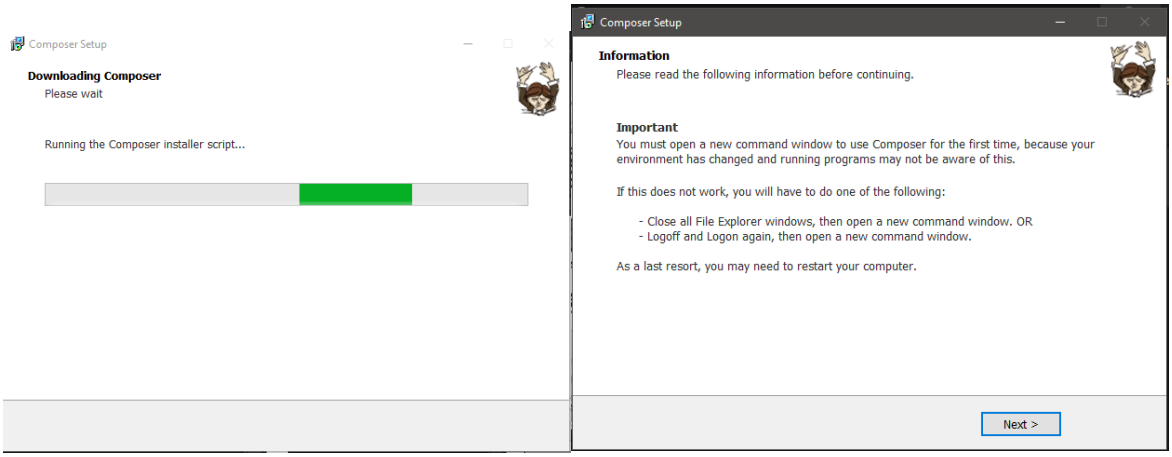
Configuración de proxy, le damos Next para completar con la instalación de Composer.

Ilustración 80 .Asignación de proxy e instalación de Composer



Completada la instalación podremos comenzar a instalar Laravel.

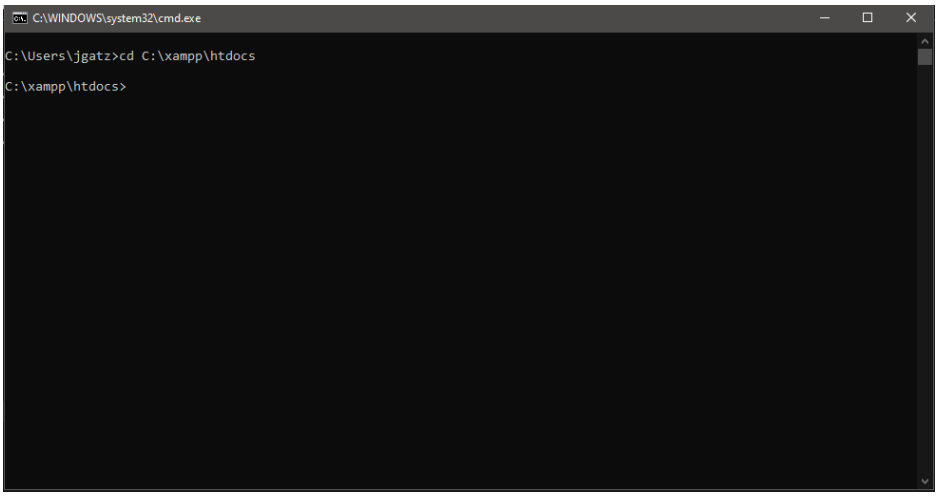
Ilustración 81. Proceso de instalación y finalización de composer



c. Laravel.

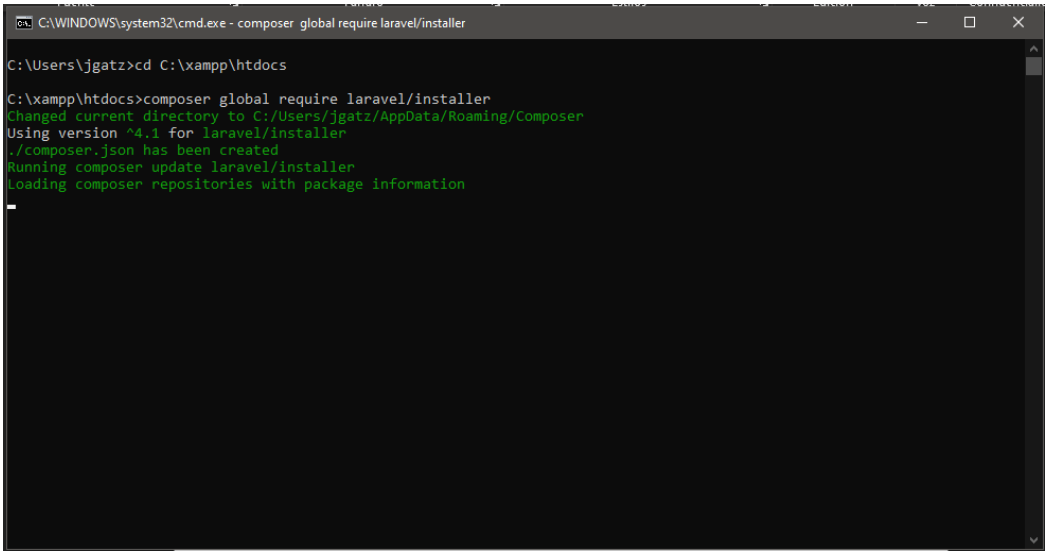
Entrar a una terminal de Windows y apuntar hacia la ruta donde se almacenan los proyectos de parte del servidor que en este caso lo es XAMPP en su ruta **C:\xampp\htdocs**

Ilustración 82. Ingresar a la ruta c:\xampp\htdocs



Estando en la ruta proseguimos a instalar Laravel mediante Composer que es el gestor de librerías de PHP con el comando **composer global require laravel/installer** que instalará todos los paquetes correspondientes.

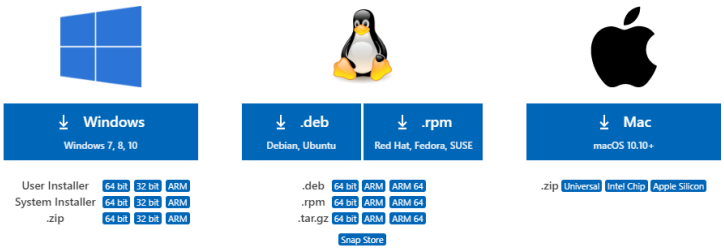
Ilustración 83. Instalación de Composer dentro del recurso del XAMPP



d. Visual Studio Code

Para descargar el ejecutable podemos ir a la dirección de descarga https://code.visualstudio.com/download.

Ilustración 84. Descargar instalador para el sistema operativo Windows



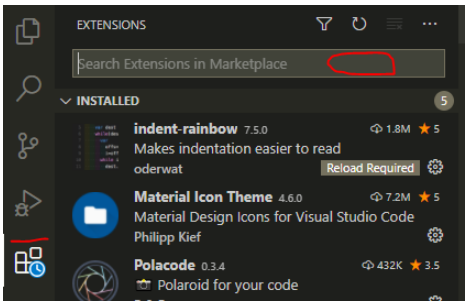
Ejecutando el instalador, se seguir los pasos del asistente de instalación como vistos con los anterior.

2. Preparación de área de trabajo.

Instalación de extensiones de Visual Studio Code

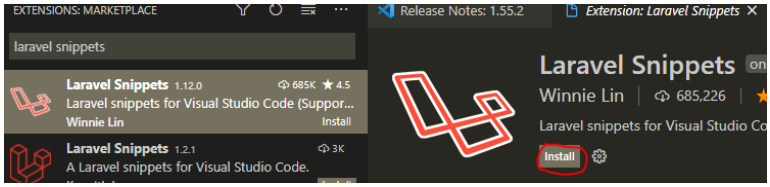
Podemos encontrar el acceso a la herramienta de extensiones en el menú ubicado del lado izquierdo y en la barra de búsqueda podemos ingresar el nombre de la extensión a instalar.

Ilustración 85. Herramienta de extensiones de Visual Studio Code



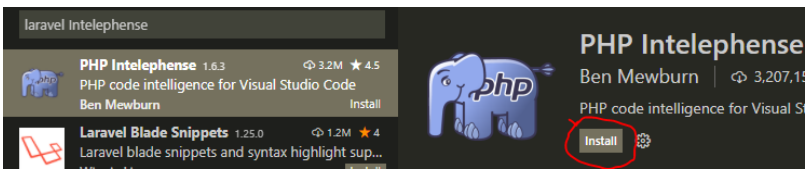
Laravel Snippets: ingresamos el nombre de la extensión y proseguimos a instalar.

Ilustración 86. Instalación de extensión Laravel Snippets



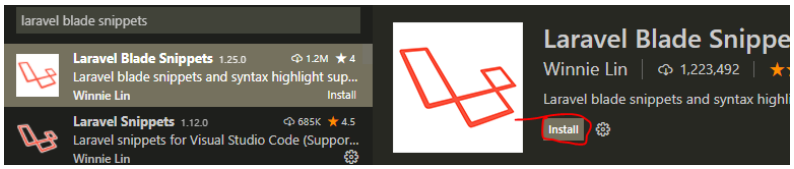
PHP Intelephense: ingresamos el nombre de la extensión y proseguimos a instalar.

Ilustración 87. Instalación de extensión PHP Intelephense



Laravel Blade snippets: ingresamos el nombre de la extensión y proseguimos a instalar.

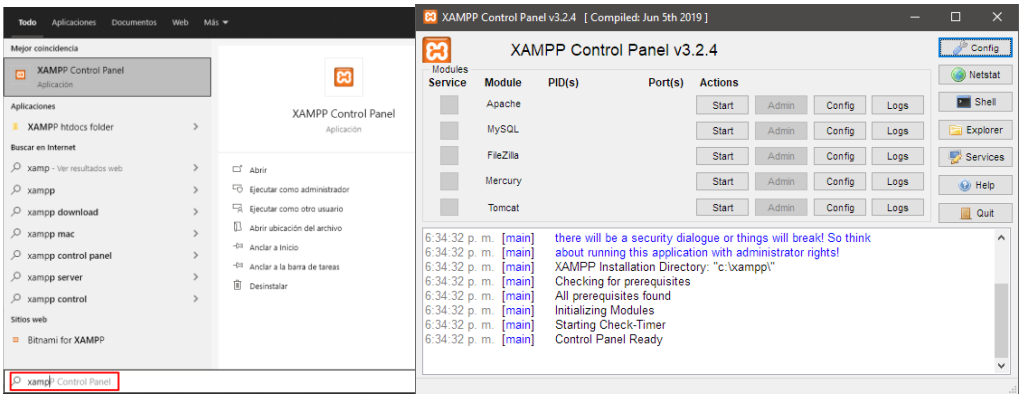
Ilustración 88. Instalación de extensión Laravel Blade Snippets



a. Inicializar servidor y MySQL

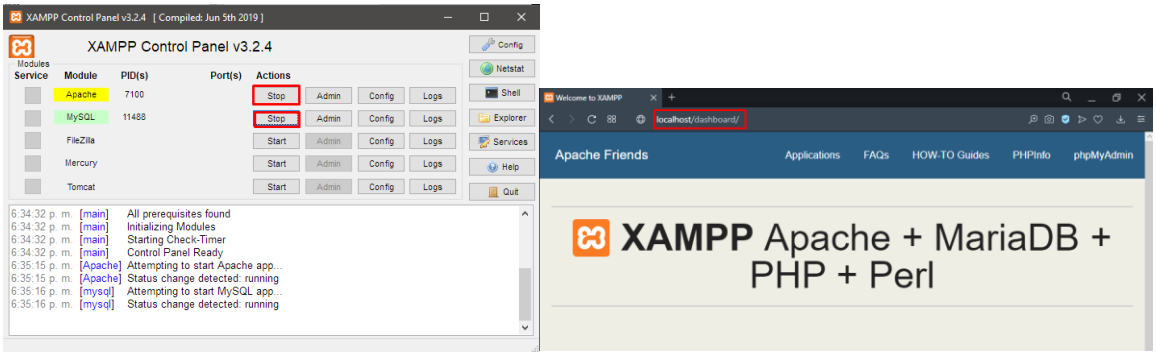
Ingresar desde el menú inicio de Windows para buscar xampp y ejecutamos el software.

Ilustración 89. Ejecutar XAMPP



Ejecutamos Apache y MySQL e ingresamos a http://localhost para verificar que se inicializó nuestro servidor.

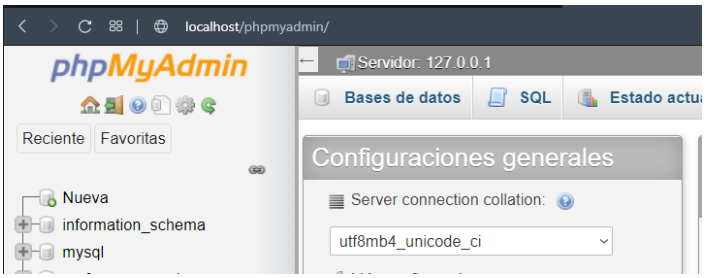
Ilustración 90. Servicio de Apache y MySQL



b. Creación de la base de datos

A través de http://localhost/phpmyadmin/ podremos ingresar al panel donde se crea la base de datos para gestionar los recursos.

Ilustración 91. Dashboard phpMyAdmin



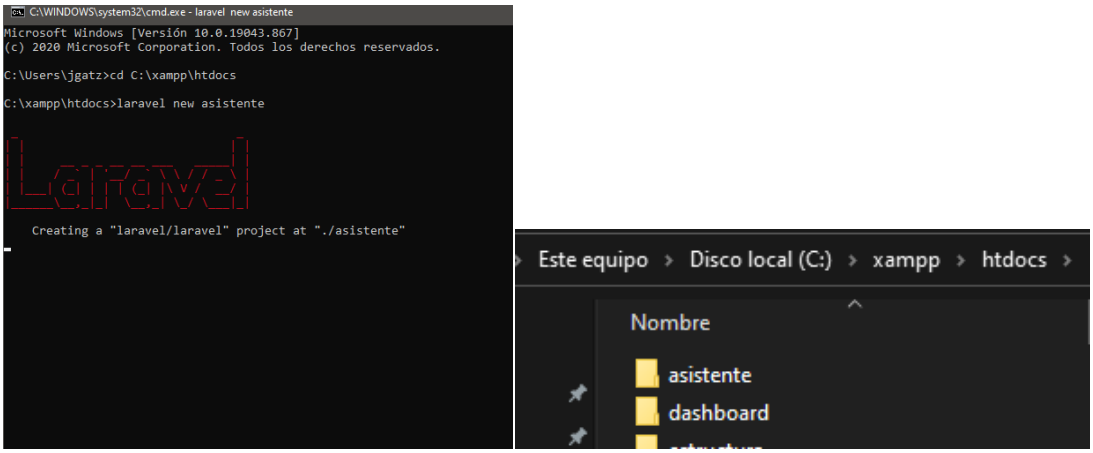
Creamos una nueva base de datos con el nombre de db_asistente

Ilustración 92. Creación de la base de datos

3. Creación del proyecto.

Estando en la ruta de **C:\xampp\htdocs** ingresamos el comando **laravel new "asistente".**

Ilustración 93. Creación del proyecto con Laravel



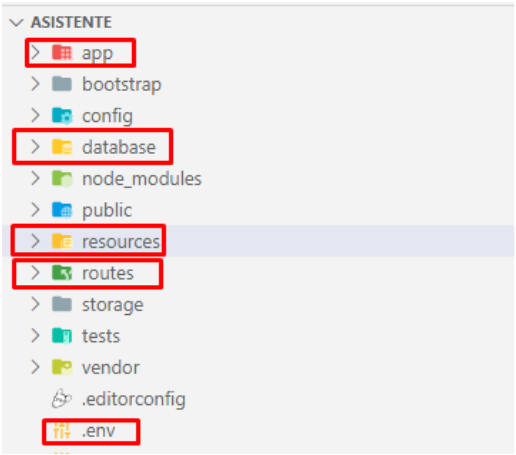
a. Estructura del proyecto

Dentro del editor de texto podemos encontrar la estructura del proyecto, teniendo los siguientes recursos relevantes que se modificó:

- **app:** donde tendremos los recursos lógicos de programación, teniendo los controladores y los modelos. Recordando que un controlador es quien intermedia entre la ruta y el modelo. El modelo es quien gestiona una tabla específica de la base de datos.
- **database:** donde se encuentra las migraciones, dichas migraciones tiene la estructura de la base de datos que se implementó.

- **public:** donde se colocan todos los recursos necesarios para la página web y que estos pueden ser visibles a los usuarios y clientes.
- **resource:** donde se crean las vistas y las plantillas que son necesaria para mostrar al usuario y cliente.
- **routes:** contiene el archivo que gestiona las peticiones del usuario y del cliente.
- **.env:** contiene la configuración para la conexión hacia la base de datos.

Ilustración 94. Estructura del proyecto asistente



b. Conexión a la base de datos.

Ingresamos al fichero .env para modificar la sección de conexión hacia MySQL, donde fue necesario cambiar el nombre de la base de datos, usuario y la contraseña que estaba configurada en http://localhost/phpmyadmin/.

Se asignó el nombre de la base de datos db_asistente, usuario root y sin contraseña ya que es para desarrollo.

Ilustración 95. Credenciales de base de datos

```
10  DB_CONNECTION=mysql
11  DB_HOST=127.0.0.1
12  DB_PORT=3306
13  DB_DATABASE=db_asistente
14  DB_USERNAME=root
15  DB_PASSWORD=
```

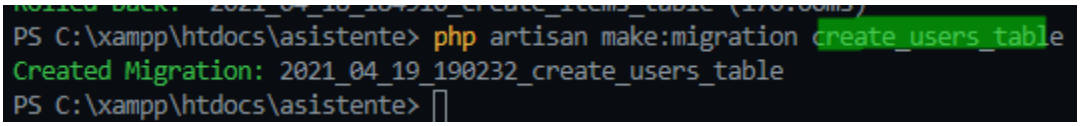
c. Migraciones

1) Creación

Fue necesario crear los ficheros de las migraciones, ingresando en la consola del editor de texto con ctrl+ñ.

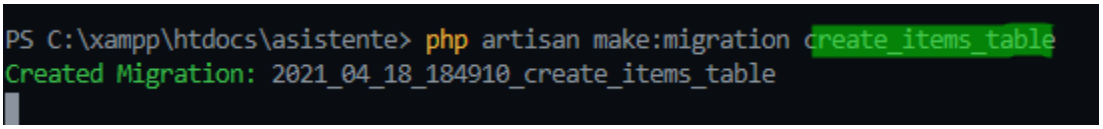
Migración de usuarios: dentro de la terminal ingresamos **php artisan make:migration** "nombre de la migración", esta apunta a la creación de una tabla en específica.

Ilustración 96. Creación de migración para usuario



Migración de productos: se ingresó la misma instrucción cambiando únicamente el nombre de la migración.

Ilustración 97. Creación de migración para productos



2) Estructura

Creada la migración se prosiguió a estructurar los atributos que llevó cada tabla, ingresando a la ruta recursos en **database->migrations** donde se encontró las dos migraciones creadas. Se modificó en la función **up()** cada una de estas al modelo de la base de datos presentada anteriormente.

Tabla usuario: se estructuró con un id, nombre, usuario, password, avatar, tipo y los atributos **rememberToken(), timestamps()** son funciones específicas de Laravel para gestionar de mejor manera las autenticaciones, la fecha de registro y actualizaciones.

Ilustración 98. Atributos para la tabla users

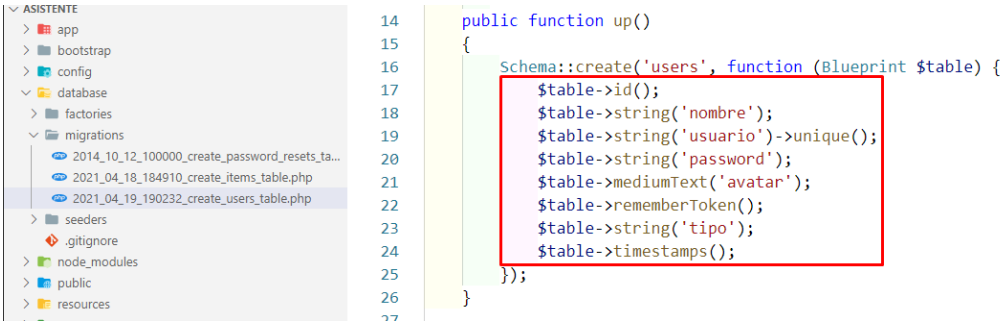
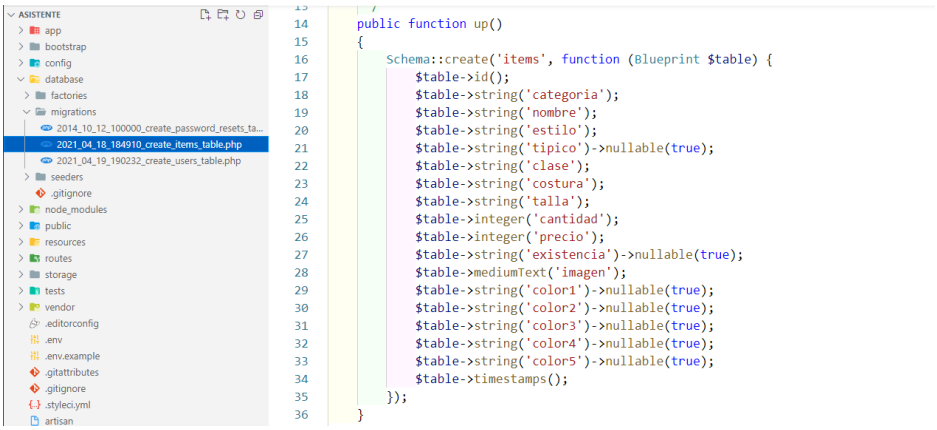


Tabla item: donde se estructuró los atributos de los productos como lo es el id, categoria, estilo, típico, clase, costura, talla, cantidad, precio, existencia, imagen, color1, color2, color3, color4, color5 y timestamps().

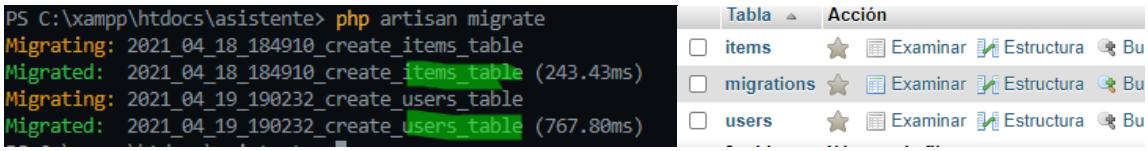
Ilustración 99. Atributos para la tabla items



3) Ejecutar migraciones

Dentro de la terminal se ejecutó el comando **php artisan migrate** para ejecutar la inserción de las tablas con sus atributos dentro de la base de datos.

Ilustración 100. Ejecutando migración



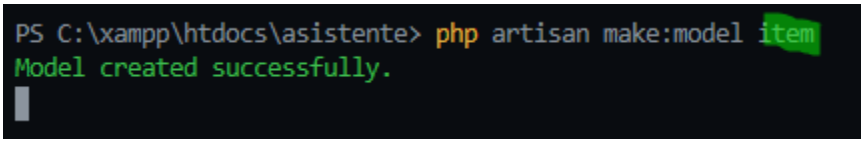
d. Modelos

Recordemos que los modelos son los que gestionan los recursos de las tablas dentro de la base de datos por ende se crea acorde al nombre de la tabla existente, para la tabla **items** se le creo un modelo con el nombre de **item** ya que se trabaja con convenciones y para la tabla **users** ya no requería de la creación ya que por defecto Laravel crea uno.

1) Creación

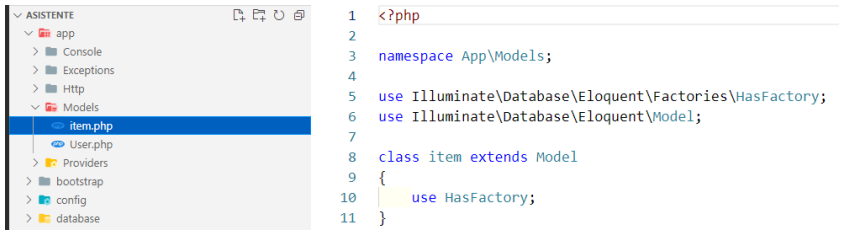
Modelo ítems: ingresando dentro de la terminal para ejecutar la instrucción de **php artisan make:model "nombre del modelo"** como lo vemos a continuación.

Ilustración 101. Creación de modelo para la tabla items



Ya creada nos dirijimos a **app->Models** y entontraremos un archivo **item.php y user.php** donde podremos instanciar los requermientos necesarios.

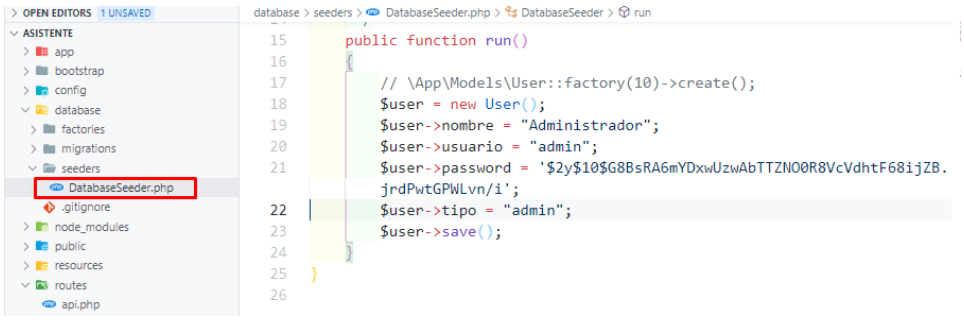
Ilustración 102. Estructura de modelo item



e. Seeders

Dentro del Seeders agregamos registro de manera estática a la tabla **User** para poder utilizar el módulo de administrador, encontramos el fichero dentro de **Database->seedersDatabaseSeeder.php**.

Ilustración 103. Ingresando credencial administrador.



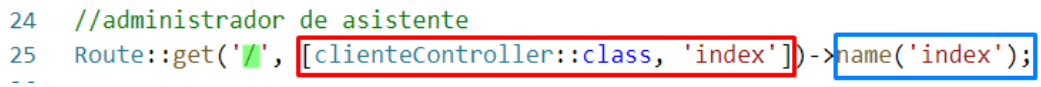
Para poder generar este registro desde la terminal se ingresó el comando **php artisan db.seed**, de esta manera se genera el credencia del usuario **admin** con contraseña **admin123**.

f. Rutas y controladores

Las rutas son los que gestionan las peticiones de los usuarios y de los clientes, se encarga de dirigir según la petición realizada a una función y estas son ejecutadas por los controladores. Estando la petición dentro del controlador puede realizar interacción con los modelos creados para ejecutar consultas y como también envía variables a las vistas.

La estructura de la ruta está de la siguiente manera:

Ilustración 104. Estructura de la ruta



Donde muestra **'/'** es la ruta al cual se puede acceder, [clienteController::class, 'index'] representan el controlador que administra esa ruta mediante una función que va entre comillas y por ultimo name('index') que es un nombre que se le asigna de manera global.

La estructura de un controlador se compone de la siguiente manera:

Ilustración 105. Estructura de una ruta

```
10      //
11      public function index(){
12          return view("cliente.index");
13      }
```

Use **App\Models\item** hace el llamado al modelo item para poder realizar acciones sobre la base de datos, **public function index()** representa la función que realiza procesos **y return view("cliente.index")** representa la acción que realiza.

La relación entre estos dos radica en que si el usuario ingresa a la ruta raíz **'/'** se envía esta petición al controlador y este retorna una vista almacenada en una carpeta **cliente** con el nombre de **index**.

g. Rutas

La rutas podemos encontrar **routes->web.php** estas rutas esta divididas en dos ya que uno está para los usuarios y otro para los clientes que necesiten acceder al asistente.

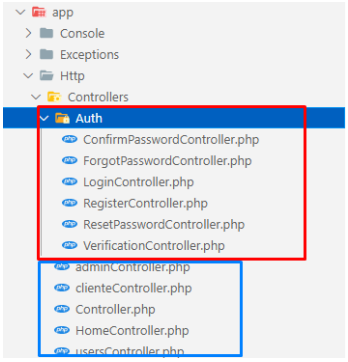
Ilustración 106. Rutas del sistema

```
web.php 1 •
routes > web.php
22  |
23  */
24  //administrador de asistente
25  Route::get('/', [clienteController::class, 'index'])->name('index');
26  Route::get('catalogo/{categoria}', [clienteController::class, 'categoria'])->name('client.showc');
27  Route::get('asistente/{id}', [clienteController::class, 'asistente'])->name('client.asist');
28
29  //administrador de admin
30  Auth::routes();
31  Route::get('loginn', [adminController::class, 'index'])->name('admin.login');
32  Route::get('usuario', [usersController::class, 'create'])->name('admin.users');
33  Route::post('usuario', [usersController::class, 'store'])->name('admin.users');
34  Route::get('usuarios', [usersController::class, 'show'])->name('admin.showu');
35  Route::get('usuario/{id}', [usersController::class, 'edit'])->name('admin.editu');
36  Route::put('usuario/{id}', [usersController::class, 'update'])->name('admin.updateu');
37  Route::get('registro', [adminController::class, 'create'])->name('admin.create');
38  Route::post('registro', [adminController::class, 'store'])->name('admin.store');
39  Route::get('inventario', [adminController::class, 'show'])->name('admin.show');
40  Route::get('editar/{id}', [adminController::class, 'edit'])->name('admin.edit');
41  Route::put('editar/{id}', [adminController::class, 'update'])->name('admin.update');
42  Route::get('consulta', [adminController::class, 'buscador'])->name('admin.buscador');
43  Route::post('consulta', [adminController::class, 'buscadorn'])->name('admin.buscadorn');
```

h. Controladores

Los controladores los encontramos en **app->Http->Controllers**, dentro de la carpeta encontramos una Auth donde se controla la validación de usuario y los demás que son los que administran las peticiones del sistema.

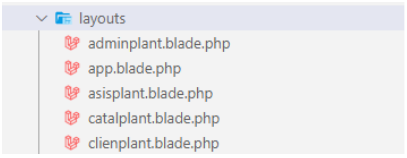
Ilustración 107. Controladores del sistema



i. Plantillas

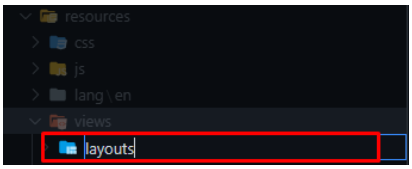
Las plantillas que se desarrolló para el sistema se encuentran dentro de **resourse->views->layouts**, dentro de esta carpeta se creó las plantillas para tener un mismo formato al entrar a las páginas.

Ilustración 108. Plantilla del sistema



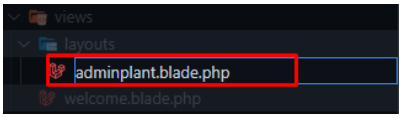
Para crear las plantillas primero debemos crear la carpeta **layouts** en de la ruta del proyecto **resources->views.**

Ilustración 109. Creación de carpeta para plantilla



Dentro de la carpeta creada anteriormente, se creó la plantilla de la siguiente forma **nombre.blade.php** la plantilla deberá llevar una extensión .blade para que sea interpretado por Blade.

Ilustración 110. Crear plantilla



Para poder agregar código dinámico dentro de la plantilla debemos identificar con @yield(“nombre de la sección”), que se puede llamar dentro de la vista para incluir código dinámico.

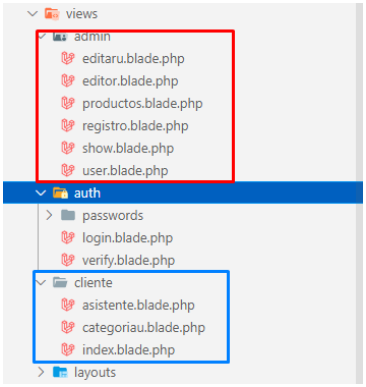
Ilustración 111. Sección dinámica para la plantilla

```
@yield('recurso')
<title>@yield('titulo')</title>
```

j. Views

Dentro de las vistas se crearon para cada uno de los requerimientos presentados por el cliente, teniendo en cuenta que se segmentó en dos módulos que son de administración y para el cliente.

Ilustración 112. Las vistas del sistema

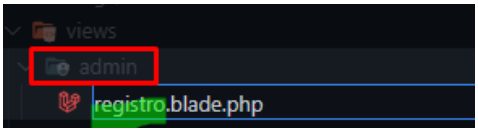


Para crear las vistas es necesario crear una carpeta por cada módulo, empezando por la de admin.

Ilustración 113. Crear carpeta para las vistas

Dentro de esta carpeta se crea la vista con la extensión .blade.

Ilustración 114. Crear una vista



Ya creada la vista hacemos el uso de la plantilla dentro de la vista de esta forma, con **@extends('nombre de la plantilla')** que es la vinculación entre la plantilla y la vista, luego hacemos el llamado de las secciones con **@section("nombre de yield") Código @endsection.**

Ilustración 115. Se extiende la plantilla dentro de la vista creada

```
1  @extends('layouts.adminplant')
2
3  @section('recurso')
4  <link href="css/admin/dropzone.css" rel="stylesheet">
5  <link href="css/admin/main.css" rel="stylesheet">
6  <link href="css/admin/cropper.css" rel="stylesheet">
7  <script src="js/admin/dropzone.js"></script>
8  <script src="js/admin/cropper.js"></script>
9  @endsection
```

4. Codificación del proyecto.

Ya presentado como se estructuró el proyecto, podemos empezar a definir las rutas, controladores, funciones, vistas y recursos utilizados dentro de cada módulo.

a. Módulo administrador

1) Login

Ruta par amostrar la página: Laravel cuenta con rutas específicas que administran las peticiones de autenticación que este llama una ruta guardada dentro de auth con el nombre de login:

Ilustración 116. Ruta para acceder al login

```
Auth::routes();
```

Login.blade.php: tenemos la ruta login de tipo POST donde se enviarán los datos del formulario y a su vez se envía de manera automática al controlador LoginController.

Ilustración 117. Vista login

```
<div class="card-body">
    <form method="POST" action="{{ route('login') }}">
        @csrf
        <div class="form-group row">
            <label for="user" class="col-md-4 col-form-label text-md-right">{{ __('Ingreser usuario') }}</label>

            <div class="col-md-6">
                <input id="user" type="text" class="form-control @error('usuario') is-invalid @enderror" name="usuario" value="{{ old('usuario') }}" rec

                @error('usuario')
                    <span class="invalid-feedback" role="alert">
                        <strong>{{ $message }}</strong>
                    </span>
                @enderror
            </div>
        </div>
        <div class="form-group row">
            <label for="password" class="col-md-4 col-form-label text-md-right">{{ __('Contraseña') }}</label>

            <div class="col-md-6">
                <input id="password" type="password" class="form-control @error('password') is-invalid @enderror" name="password" required autocomplete=

                @error('password')
                    <span class="invalid-feedback" role="alert">
                        <strong>{{ $message }}</strong>
                    </span>
                @enderror
            </div>
        </div>
        <div class="form-group row">
            <div class="col-md-6 offset-md-4">
                <div class="form-check">
                    <input class="form-check-input" type="checkbox" name="remember" id="remember" {{ old('remember') ? 'checked' : '' }}>

                    <label class="form-check-label" for="remember">
                        {{ __('Permanecer conectado') }}
                    </label>
                </div>
            </div>
        </div>
        <div class="form-group row mb-0">
            <div class="col-md-8 offset-md-4">
                <button type="submit" class="btn btn-primary">
                    {{ __('Ingresar') }}
                </button>
            </div>
        </div>
```

Medida de seguridad

Ruta de envío de datos

LoginController: se encarga de utilizar las funciones de middleware para autenticar los datos.

Ilustración 118. Controlador de login

```
use AuthenticatesUsers;

/**
 * Where to redirect users after login.
 *
 * @var string
 */
protected $redirectTo = RouteServiceProvider::HOME;

/**
 * Create a new controller instance.
 *
 * @return void
 */
public function __construct()
{
    $this->middleware('guest')->except('logout');
}
```

Si el usuario esta autenticado

Se cea una nueva intancia para logear

Middleware: se encarga de validar los datos, sí los datos son incorrectos esta redirige a la ruta login donde se encuentra la vista login, para reintentar ingresar de nuevo al sistema.

Ilustración 119. Autenticación por Middleware

```
protected function redirectTo($request)
{
    if (! $request->expectsJson()) {
        return route('login');
    }
}
```

2) Registro de producto

Se crearon dos rutas para administrar una que es para mostrar la página con el formulario de tipo GET y la otra de tipo POST quien se encarga de tomar los datos del formulario para enviar al controlador respectivo.

Ruta para mostrar página: tiene una ruta denominada admin.create que se encarga de enviar la petición hacia adminController a la función create.

Ilustración 120. Ruta para acceder a registrar un producto

```
Route::get('registro', [adminController::class, 'create'])->name('admin.create');
```

adminController: la función create llama a la vista registro, que contiene el formulario para registrar el producto

Ilustración 121. Función create que administra la ruta

```
public function create(){
    return view("admin.registro");
}
```

Registro.blade.php: se llama a la plantilla adminplant, dentro de este se agrega el formulario donde se ingresan los registros, se carga una imagen en formato PNG, cuenta con un modal que ayuda a recortar la imagen que será subida en el formulario y se agregaron los scripts correspondiente de las librerías.

Ilustración 122. Vista registrar producto

```
resources > views > admin > registro.blade.php > script
@extends('layouts.adminplant')

@section('recurso')
<link href="css/admin/dropzone.css" rel="stylesheet">
<link href="css/admin/main.css" rel="stylesheet">
<link href="css/admin/cropper.css" rel="stylesheet">
<script src="js/admin/dropzone.js"></script>
<script src="js/admin/cropper.js"></script>
@endsection

@section('titulo', 'registro')

@section('contenido')

<!--Modal de editor de imagen-->
  <form class="row text-center" action="{{route('admin.store')}}" method="POST">···
  </form>

<!--Modal de editor de imagen-->
  <div class="modal fade vh-90" id="modal" tabindex="-1" role="dialog" aria-labelledby="modalLabel" aria-hidden="true">···
  </div>
@endsection

@section('js')
<script src="js/admin/vibrant.js"></script>
<script src="js/admin/app.js"></script>
@endsection
```

app.js: donde se encuentra la función para tomar las muestras de colores a la imagen que es subida en el formulario, contiene el evento de onchange para determinar el cambio dentro del input tipo file, de ser cargado la imagen entra en acción Vibrant mediante la función **getColor()**.

Ilustración 123. Script para tomar la muestra de color de la imagen cargada

```
$(document).ready(function(){

    var $modal = $('#modal'),
    image = document.getElementById('sample_image'),
    cropper;

    let imgdata = document.getElementById('uploaded_image'),
    paletteReady = false,
    list = document.getElementById('colors'),
    idimg = document.getElementById('inmGombre');
    //evento al cargar una imagen
    $('#upload_image').change(function(event){···
    });
    //modal para editar la imagen agregada
    $modal.on('shown.bs.modal', function() {···
    }).on('hidden.bs.modal', function(){···
    });
    $('#cancel').click(function(){···
    });
    $('#crop').click(function(){···
    });
    //funcion que toma el color de la imagen recortada
    function getColor(img) {···
    }
    //eliminar objeto.
    function cleanObj(obj){···
    }
    });
```

Ruta enviar datos: dentro del form se encuentra un método que ayuda enviar los datos y está la envía hacia una ruta **admin.store** que se encarga de registrar los datos.

Ilustración 124. Ruta que administra los datos enviados desde el formulario

```
Route::post('registro', [adminController::class, 'store'])->name('admin.store');
```

adminController: la función store valida los campos que no estén nulos e instancia al modelo **item** para poder realizar acciones sobre la base de datos. La función **save()** guarda los datos y por ultimo **redirect()** que redirige a una ruta de ser realizado la inserción.

Ilustración 125. Función store que administra la ruta y los datos enviados

```
public function store(Request $data){
    $data->validate([
        'image' => 'required',
        'categoria' => 'required',
        'nombre' => 'required',
        'estilo' => 'required',
        'talla' => 'required',
        'cantidad' => 'required',
        'precio' => 'required'
    ]);
    $item = new item();
    $item->nombre = $data->nombre;
    $item->categoria = $data->categoria;
    $item->estilo = $data->estilo;
    $item->talla = $data->talla;
    $item->clase = $data->clase;
    $item->costura = $data->costura;
    $item->tipico = $data->tipico;
    $item->imagen = $data->image;
    $item->existencia = "1";
    $item->precio = $data->precio;
    $item->cantidad = $data->cantidad;
    $item->color1 = $data->Vibrant;
    $item->color2 = $data->Muted;
    $item->color3 = $data->DarkVibrant;
    $item->color4 = $data->DarkMuted;
    $item->color5 = $data->LightVibrant;
    $item->save();
    return redirect()->route('admin.create');
    }
```

3) Inventario

Muestra todos los productos registrados en el sistema, los productos se enlistan gracias a los cards de Bootstrap y donde se puede realizar la actualización del mismo.

Ruta para mostrar página: tiene una ruta denominada admin.show que se encarga de enviar la petición hacia adminController a la función show.

Ilustración 126. Ruta para acceder a inventario

```
Route::get('inventario', [adminController::class, 'show'])->name('admin.show');
```

adminController: la función show realizará una consulta al modelo de todos los productos existentes y luego llama a la vista productos y envía la consulta previa realizada.

Ilustración 127. Función show que administra la ruta y realiza una consulta

```
public function show(Request $request){
    $request = $request->get('buscar');
    $items = item::where('existencia', '1')->where('nombre', 'like', $request.'%')->get();
    return view('admin.productos', compact('items'));
}
```

producto.blade.php: se llama a la plantilla adminplant, se recorre la variable items con un foreach dándole un formato de card por cada producto encontrado y segmentando por categoría, dentro de los card se agregó un botón para modificar el producto. De la misma manera en la página se agregó un input para consultar los productos por nombre.

Ilustración 128. Vista del inventarios

```
@section('contenido')
<nav class="navbar navbar-light bg-light text-end">···
</nav>                                                    Busqueda
<h2>Blusas</h2>
    <div class="row row-cols-auto">    Recorrer consula enviada desde controlador
        @foreach ($items->all() as $item)
        @if ($item->categoria == "Blusas")
        <div class="col-auto">
            <div class="card shadow-sm">
                <div class="card card-cover h-100 overflow-hidden text-white bg-light rounded-5">···
                </div>
                <div class="card-body bg-transparent">
                    <h5 class="card-title">{{$item->nombre}}</h5>
                    <p class="card-text"><small class="text-muted">···
                    </small></p>
                    <div class="d-flex justify-content-end">
                        <div class="btn-group-sm text-end">                    Editar producto
                            <a href="{{route('admin.edit', $item->id)}}" class="btn btn-sm btn-outline-secondary">Editar</a>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        @endif
        @endforeach
    </div>
<hr>
```

4) Actualizar producto

Ruta enviar id: dentro del card se encuentra un vínculo donde se envía el id hacia la ruta **admin.edit,** que se encarga de pasar hacia la función edit del controlador.

Ilustración 129. Ruta para acceder a editar producto

```
Route::get('editar/{id}', [adminController::class, 'edit'])->name('admin.edit');
```

adminController: la función edit realiza una consulta del id y lo envía hacia la vista editor.blade.php.

Ilustración 130. Función que administra la ruta y realiza una consulta de un producto

```
    public function edit($id){
        $item = item::find($id);
        return view('admin.editor', compact('item'));
    }
```

editor.blade.php: se muestra los datos de la variable item dentro del formulario descrito en la vista registro, ya que se reutilizó el mismo modelo y solo se agregó un método @method('PUT'), para enviar los datos de esta manera se podrá actualizar los datos del producto.

Ilustración 131. Vista editor de producto

```
@section('contenido')
    <form class="row text-center" action="{{route('admin.update', $item)}}" method="POST">
        @csrf
        @method('PUT')
            <div class="text-center">
                <label for="upload_image">
                    <img src="{{$item->imagen}}" id="uploaded_image" class="rounded mx-auto d-block" height="150"/>
                    <textarea id="inmGombre" name="image" rows="4" cols="50" style="display: none;" required>{{$item->imagen}}</texta
                    <div class="">
                        <div class="text">Cambiar foto</div>
                    </div>
                    <input type="file" class="image" id="upload_image" style="display:none" accept="image/png"/>
                    <div class="row" id="colors">
                        <input value="{{$item->color1}}" name="Vibrant" class="col form-control m-1" style="color:{{$item->color1}};
                        <input value="{{$item->color2}}" name="Muted" class="col form-control m-1" style="color:{{$item->color2}}; ba
                        <input value="{{$item->color3}}" name="DarkVibrant" class="col form-control m-1" style="color:{{$item->color3
                        <input value="{{$item->color4}}" name="DarkMuted" class="col form-control m-1" style="color:{{$item->color4}}
                        <input value="{{$item->color5}}" name="LightVibrant" class="col form-control m-1" style="color:{{$item->color
                    </div>
```

Metodo de envio de datos

Ruta

Mostrar datos de item

Ruta enviar datos: la ruta que administra los datos para actualizar los datos.

Ilustración 132. Ruta que administra los datos enviados desde el formulario para actualizar

```
Route::put('editar/{id}', [adminController::class, 'update'])->name('admin.update');
```

adminController: la función update toma los datos enviados desde la ruta para poder validar los datos y realizar la actualización, después de realizar la actualización redirecciona a la ruta admin.show.

Ilustración 133. Función update para realizar actualización del producto

```
public function update(Request $udata, $id)
{
    $udata->validate([
        'image' => 'required',
        'categoria' => 'required',
        'nombre' => 'required',
        'estilo' => 'required',
        'talla' => 'required',
        'cantidad' => 'required',
        'precio' => 'required'
    ]);

    $item = item::find($id);
    $item->id = $id;
    $item->nombre = $udata->nombre;
    $item->categoria = $udata->categoria;
    $item->estilo = $udata->estilo;
    $item->talla = $udata->talla;
    $item->clase = $udata->clase;
    $item->costura = $udata->costura;
    $item->tipico = $udata->tipico;
    $item->imagen = $udata->image;
    $item->precio = $udata->precio;
    $item->cantidad = $udata->cantidad;
    $item->existencia = $udata->existe;
    $item->color1 = $udata->Vibrant;
    $item->color2 = $udata->Muted;
    $item->color3 = $udata->DarkVibrant;
    $item->color4 = $udata->DarkMuted;
    $item->color5 = $udata->LightVibrant;
    $item->save();
    return redirect()->route('admin.show');
}
```

Valida los datos enviados desde la ruta

Actualiza los datos

Redirecciona a la ruta

5) Registrar usuario

Ruta para mostrar página: está identificada con el nombre de admin.users que se encarga de enviar la petición hacia usersController a la función create.

Ilustración 134. Ruta para acceder a registrar usuario

```
Route::get('usuario', [usersController::class, 'create'])->name('admin.users');
```

userController: la función create retorna la vista user para mostrar el formulario de registro.

Ilustración 135. Función que administra la ruta

```php
public function create(){
    return view('admin.user');
}
```

User.blade.php: emplea el mismo formulario que se vio en registro y los datos que se ingresen dentro del formulario se enviará a la ruta admin.users con un método POST.

Ilustración 136. Vista crear usuario

```
@section('contenido')
<div class="col-8 m-auto justify-content-md-center">
<form class="row text-center" action="{{route('admin.users')}}" method="POST">
    @csrf
      <div class="text-center p-3">
        <label for="upload_image">
          <img src="img/camara.png" id="uploaded_image" class="rounded mx-auto d-block" height="100"/>
          <textarea id="inmGombre" name="image" rows="4" cols="50" style="display: none;"></textarea>
          <div class="">
            <div class="text">Subir foto</div>
          </div>
          <input type="file" class="image" id="upload_image" style="display:none" accept="image/*" required/>
        </label>
      </div>
```

Ruta enviar datos: los datos enviados desde el formulario son procesados por la clase store del usersController, donde se encarga de realizar la inserción necesaria.

Ilustración 137. Ruta que administra los datos enviados del formulario

```php
Route::post('usuario', [usersController::class, 'store'])->name('admin.users');
```

userController: en la función store se encarga de recibir los datos del formulario para luego validar y realizar el registro correspondiente.

Ilustración 138. Función store que administra la ruta y los datos enviados

```php
public function store(Request $data){

    $data->validate([
        'usuario' => 'required|unique:users',
        'password' => 'required|min:8|confirmed'
    ]);

    $user = new User();
    $user->nombre = $data->nombre;
    $user->usuario = $data->usuario;
    $user->password = Hash::make($data->password);
    $user->avatar = $data->image;
    $user->tipo = $data->tipouser;
    $user->save();
    return redirect()->route('admin.showu');
}
```

6) Ver usuarios

Ruta para mostrar página: tiene una ruta denominada admin.showu que se encarga de enviar la petición hacia usersController a la función show.

Ilustración 139. Ruta para acceder a ver usuarios

```
Route::get('usuarios', [usersController::class, 'show'])->name('admin.showu');
```

usersController: la función show llama a la vista show para mostrar todos los usuarios registrados dentro del sistema.

Ilustración 140. Función que administra la ruta y realiza la consulta de todos los usuarios

```
public function show(){
    $users= User::all();
    return view('admin.show', compact('users'));
}
```

show.blade.php: se llama a la plantilla adminplant, dentro de este se agrega el formulario donde se ingresan los datos, un modal que ayuda a recortar la imagen que es subida en el formulario y como también se agregaron los scripts correspondientes de las librerías.

Ilustración 141. Vista usuarios

```
@extends('layouts.adminplant')

@section('titulo', 'Usuarios')

@section('contenido')
<table class="table ">
    <thead>
      <tr>
        <th scope="col">#</th>
        <th scope="col">Usuario</th>
        <th scope="col">Nombre</th>
        <th scope="col"></th>
        <th scope="col"></th>
      </tr>
    </thead>
    <tbody>                                   Se recorre la consulta
      @foreach ($users as $user)
      <tr class="align-middle">
          <th scope="row">{{$user->id}}</th>
          <td>{{$user->usuario}}</td>
          <td>{{$user->nombre}}</td>
          <td>
              <img class="rounded-circle me-2" src="{{$user->avatar}}" width="150" height="150"/>
          </td>
          <td>
              <div class="btn-group-sm text-end">
                  <a href="{{route('admin.editu', $user->id)}}" class="btn btn-sm btn-outline-secondary">Editar</a>
              </div>
          </td>                               Ruta para enviar id
        </tr>                                 para actualizar
      @endforeach
    </tbody>
  </table>
@endsection
```

Ruta enviar id: dentro de la ruta **admin.editu** se envía el id del usuario para ser actualizado.

Ilustración 142. Ruta para acceder a editar usuario

```
Route::get('usuario/{id}', [usersController::class, 'edit'])->name('admin.editu');
```

usersController: la función edit realiza la consulta de los datos del usuario, para luego enviar a la vista editaru.

Ilustración 143. Función edit que administra la ruta y realiza una consulta

```
public function edit($id){
    $user = User::find($id);
    return view('admin.editaru', compact('user'));
}
```

Editaru.blade.php: muestra los datos de la consulta realizada previamente dentro del formulario y en el mismo se encuentra la ruta donde se envía los datos a través del método @method('PUT') a la ruta admin.updateu.

Ilustración 144. Vista editar usuario

```
@section('contenido')
    <div class="col-8 m-auto justify-content-md-center">
    <form class="row text-center" action="{{route('admin.updateu', $user)}}" method="POST">
        @csrf
        @method('PUT')
        <div class="text-center p-3">
            <label for="upload_image">
                <img src="{{$user->avatar}}" id="uploaded_image" class="rounded mx-auto d-block" height="200"/>
                <textarea id="inmGombre" name="image" rows="4" cols="50" style="display: none;">{{$user->avatar}}</textarea>
                <div class="">
                    <div class="text">Subir foto</div>
                </div>
                <input type="file" class="image" id="upload_image" style="display:none" accept="image/*"/>
            </label>
        </div>
```

Ruta donde se envia los datos actualizados

Muestra los datos del usuario

Ruta enviar datos: la ruta admin.updateu se envía los datos del formulario anterior hacia la función update en el controlador usersController.

Ilustración 145. Ruta para actualizar usuario

```
Route::put('usuario/{id}', [usersController::class, 'update'])->name('admin.updateu');
```

usersControllers: la función update se encarga de validar los campos, realiza la actualización del mismo y redirige a la ruta admin.showu.

Ilustración 146. Función update que actualiza datos de usuario

```
public function update(Request $datau, $id){
    $datau->validate([
        'usuario' => 'required',
        'password' => 'required|min:8|confirmed'
    ]);
    $user = User::find($id);
    $user->nombre = $datau->nombre;
    $user->usuario = $datau->usuario;
    $user->password = Hash::make($datau->password);
    $user->avatar = $datau->image;
    $user->tipo = $datau->tipouser;
    $user->save();
    return redirect()->route('admin.showu');
}
```

Valida los datos

Actualiza

b. Asistente

Ruta para mostrar página: tiene una ruta denominada index que se encarga de enviar la petición hacia clienteController a la función index.

Ilustración 147. Ruta para acceder a página inicio

```
//administrador de asistente
Route::get('/', [clienteController::class, 'index'])->name('index');
```

clienteController: la función index llama a la vista index creada dentro de la carpeta cliente.

Ilustración 148. Función index que administra la ruta

```
public function index(){
    return view("cliente.index");
}
```

Index.blade.php: contiene la ruta para inicio de sesión al sistema, el menú de categoría donde se envía a la ruta el tipo de categoría que seleccionó por el cliente.

Ilustración 149. Ruta para acceder al login

```
@section('contenido')
    <!--Bienvenida-->
    <div class="row" id="DivBien">
        <div class="d-flex justify-content-end">
            <a href="{{route('login')}}" class="btn btn-sm btn-outline-secundary">Ingresar</a>
        </div>
```

Ingresar al modulo administrador

Ilustración 150. Ruta para ir a catálogo

```
<div class="row row-cols-2 vh-100">
    <label class="checkeable col p-2 text-end">
        <a href="{{route('client.showc', 'Blusas')}}">
        <div class="card card-cover text-end h-100 overflow-hidden bg-white rounded-5 shadow-sm">
            <img class="position-absolute" src="img/blusa.png"/>
            <div class="position-absolute d-flex flex-column h-100 p-5 pb-3 text-shadow-1">
                <h2 class="pt-5 mt-5 mb-4 display-6 lh-1 fw-bold"><small class="text-muted">Blusas</small></h2>
            </div>
            <input class="form-check-input" type="radio" name="item1"/>
        </div>
        </a>
    </label>
```

Ruta para mostrar página: la ruta client.showc toma el valor de la categoría seleccionada y lo envía a la función categorias el controlador clienteController.

Ilustración 151. Ruta para acceder a categoría

```
Route::get('catalogo/{categoria}', [clienteController::class, 'categoria'])->name('client.showc');
```

clientController: toma el valor de la categoría seleccionada por el cliente y realiza una consulta limitando por la categoría. Enviando la consulta hacia la vista categoriau

Ilustración 152. Función categoria para administrar la ruta y realizar una consulta

```
public function categoria($categoria){
    $items = item::where('categoria', '=', $categoria)->where('existencia', '=', 1)->get();
    return view('cliente.categoriau', compact('items', 'categoria'));
}
```

categoriasu.blade.php: recorre la variable de la consulta realizada y muestra los datos en un formato de card, cuenta con una ruta para referir los productos similares según los colores que tiene el producto seleccionado.

Ilustración 153. Vista categoría

```
@section('contenido')
<div class="row row-cols-auto">
    @foreach ($items as $item)
    @if ($item->id != null)
    <div class="col-auto">
        <div class="card shadow-sm">
            <div class="card card-cover h-100 overflow-hidden text-white bg-light rounded-5">
                <img class="car-image" src="{{$item->imagen}}">
            </div>
            <div class="card-body bg-transparent">
                <h5 class="card-title">{{$item->nombre}}</h5>
                <p class="card-text"><small class="text-muted">
                    <b>Estilo:</b> {{$item->estilo}}
                    <b>Precio:</b> Q {{$item->precio}}
                </small></p>
                <div class="d-flex justify-content-end">
                    <div class="btn-group-sm text-end">
                        <a href="{{route('client.asist', $item)}}" class="btn btn-sm btn-outline-secondary">Completar con traje</a>
                    </div>
                </div>
            </div>
        </div>
    </div>
    @else
        <h2>Sin articulos para mostrarsa</h2>
    @endif
```

Envia el id del producto para buscar coincidencia

c. Referencia

Ruta para mostrar página: tiene una ruta denominada client.asist que se encarga de enviar el id del producto seleccionado a clienteController a la función asistente.

Ilustración 154. Ruta para acceder referencia

```
Route::get('asistente/{id}', [clienteController::class, 'asistente'])->name('client.asist');
```

clienteController: la función asistente realiza una consulta a los colores del producto seleccionado, luego realiza una consulta a los atributos id, color1, color2, color3, color4, color5, imagen, nombre y precio, compara las coinciden con la consulta anterior y lo devuelve en la vista asistente.

Ilustración 155. Función asistente que realiza la comparación de colores con otros productos

```
public function asistente($id){
    $query = item::select('id','color1','color2','color3','color4','color5','categoria')->find($id);
    $tquery = item::select('id','color1','color2','color3','color4','color5','categoria','imagen','nombre','precio')
        ->orWhere('color1', $query->color1)
        ->orWhere('color2', $query->color1)
        ->orWhere('color3', $query->color1)
        ->orWhere('color4', $query->color1)
        ->orWhere('color5', $query->color1)
        ->orWhere('color1', $query->color2)
        ->orWhere('color2', $query->color2)
        ->orWhere('color3', $query->color2)
        ->orWhere('color4', $query->color2)
        ->orWhere('color5', $query->color2)
        ->orWhere('color1', $query->color3)
        ->orWhere('color2', $query->color3)
        ->orWhere('color3', $query->color3)
        ->orWhere('color4', $query->color3)
        ->orWhere('color5', $query->color3)
        ->orWhere('color1', $query->color4)
        ->orWhere('color2', $query->color4)
        ->orWhere('color3', $query->color4)
        ->orWhere('color4', $query->color4)
        ->orWhere('color5', $query->color4)
        ->orWhere('color1', $query->color5)
        ->orWhere('color2', $query->color5)
        ->orWhere('color3', $query->color5)
        ->orWhere('color4', $query->color5)
        ->orWhere('color5', $query->color5)
        ->get();
    return view('cliente.asistente', compact('tquery', 'query'));
}
```

Consulta del del producto selecionado

Consulta a todos los productos registrados en el sistema

asistente.blade.php: dentro de la vista, se realiza un recorrido de la consulta realizada para segmentar por categorías agregándolo dentro de un carouser de Bootstrap.

Ilustración 156. Vista referencia y muestra datos según categoría

```
<div class="row">
@php
    $val = 1;
    foreach($tquery as $item){
        if ($item->categoria == "Blusas") {
            if ($val <= 4) {
            echo '<div class="col-6 col-sm-3">
                    <div class="card">
                        <img src="'.$item->imagen.'" class="card-img-top">
                    </div>
                    <div class="card-body">
                        <h6 class="text-uppercase fw-light text-center">'.$item->nombre.'</h6>
                    </div>
                </div>';
            }
        }

        $val++;
    }
@endphp
```

Limitacion por categoria

Paginacion

Dentro el mismo, pero en un apartado se muestra la primera coincidencia que pudo encontrar en una vista previa que tiene. El recorrido de la consulta lo ase de la siguiente manera, sí ya se mostró un producto se finaliza el mismo.

Ilustración 157. Vista previa de un producto

```
<div class="b" style="max-height: 37vh">
@foreach ($tquery as $it)
@if ($it->categoria == "Blusas")
    <img src="{{$it->imagen}}" id="blusaView">
    @break
@endif
@endforeach
```

js: si el cliente da clic sobre algunas de las referencias encontradas, la imagen cambie en la vista previa.

Ilustración 158. Script para controlar selección de imagen

```
$('#caroBlusa').on('click', 'img', function(){
var thisSrc = $(this).attr('src');
imgBlusa.src = thisSrc;
});
$('#caroFaja').on('click', 'img', function(){
var thisSrc = $(this).attr('src');
imgFaja.src = thisSrc;
});
$('#carCortes').on('click', 'img', function(){
var thisSrc = $(this).attr('src');
imgCorte.src = thisSrc;
});
$('#caroZapatos').on('click', 'img', function(){
var thisSrc = $(this).attr('src');
imgZapato.src = thisSrc;
});
```

5. Seguridad.

Dentro del sistema se trabajó la autenticación propia de Laravel con la función de Auth(), esta función limita el accesos haca las rutas del módulo de administración, todo petición que es realizada dentro de las rutas del módulo deberá ser evaluado, para los formulario se agregaron un roten para validar los formulario a través del @csrf.

Ilustración 159. Ingresamos a la ruta de registro de producto sin haber ingresado credencial.

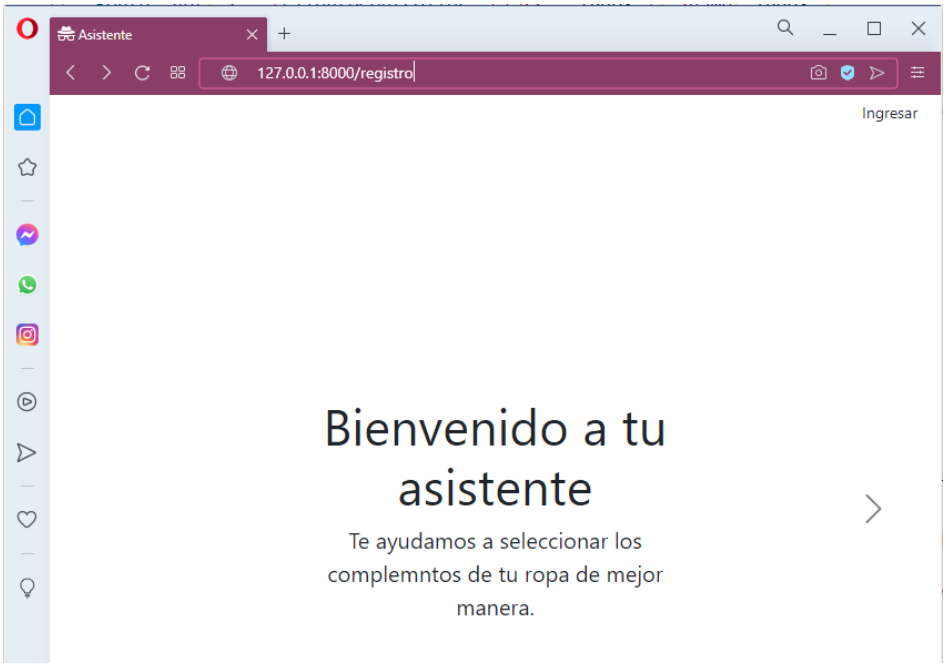
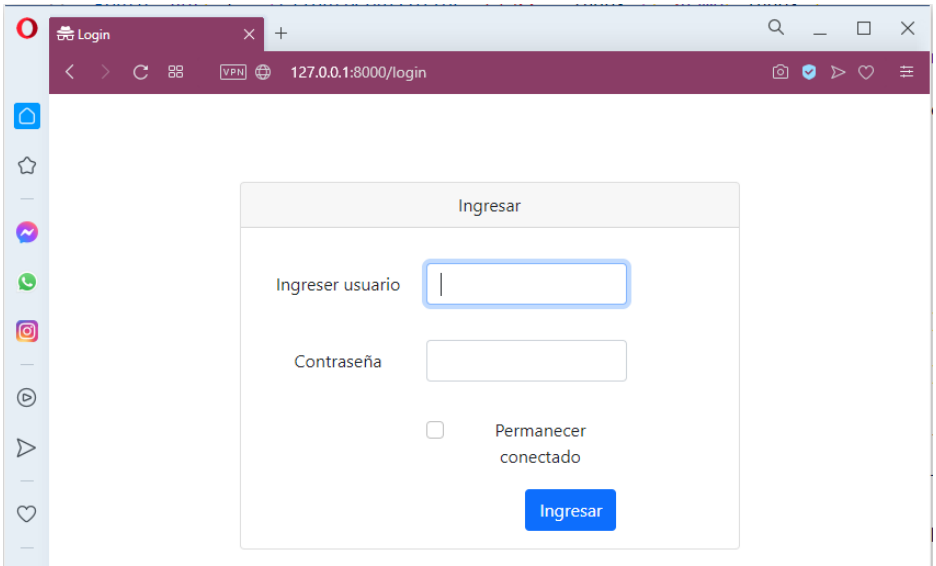


Ilustración 160. Al realizar la consulta la función Auth valida si el usuario se encuentra activo en la sección de lo contrario pregunta las credenciales correspondientes.



6. Repositorio del proyecto

Ingresamos en el siguiente link https://github.com/TheRevGt/asistente, para poder descargar los recursos y clonar el repositorio dentro del servidor.

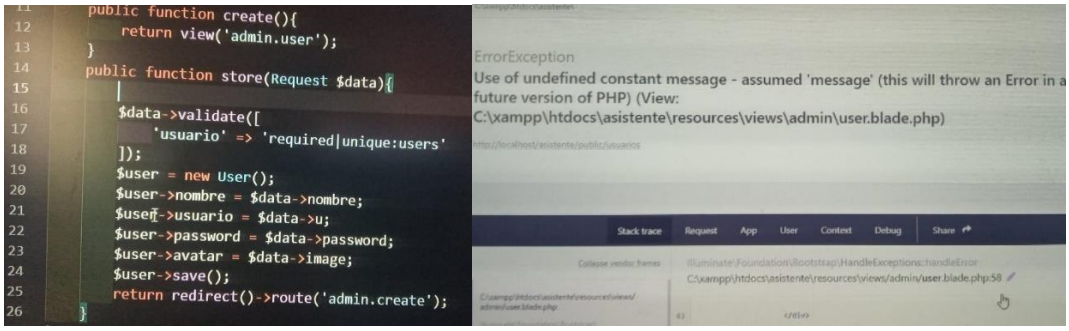
F. EVOLUCIÓN DEL PROTOTIPO

La evolución del prototipo se centró en dos áreas siendo estas la del interfaz de usuario y la lógica de programación.

1. Fase de pruebas.

Gracias a las pruebas de estrés empleadas dentro del desarrollo del proyecto. se logró detectar las fallas dentro del registro de datos, validación de campos de formularios, validación de campos para el registro dentro de la base de datos y el área de seguridad como se vio en el apartado anterior.

Ilustración 161. Prueba de estrés, validación de campos de formulario.

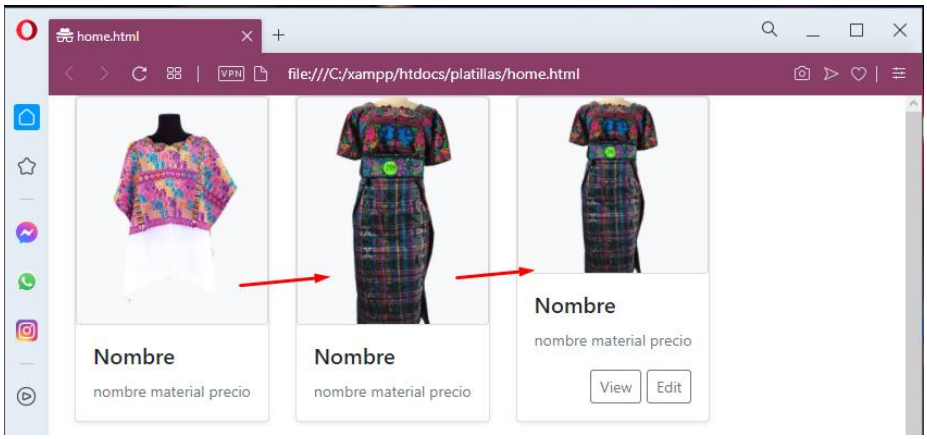


2. Interfaz de usuario.

Para el interfaz de usuario se empleó Bootstrap que ayudó a modelar el diseño de los formularios, menús, tarjetas o conocidas como Cards, entre otras más que se fue acoplando según las necesidades requeridas como lo veremos a continuación.

Creación de card: se empezó desde un modelo básico hasta tener el modelo que se está implementando en el sistema actualmente.

Ilustración 162. Evolución de Card para mostrar producto



Menú de navegación: se modeló en base a la función RESPONSIVE, en éste modelo se vio que el menú propuesto no cumplía con las funcionalidades del sistema y fue necesario modificar.

Ilustración 163. Menú anterior

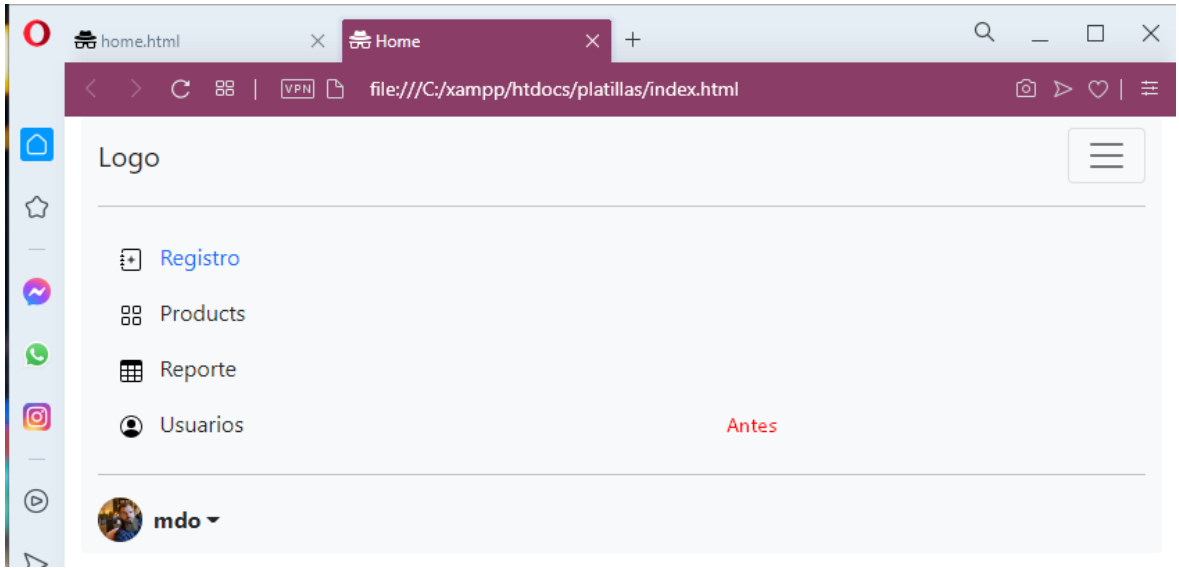
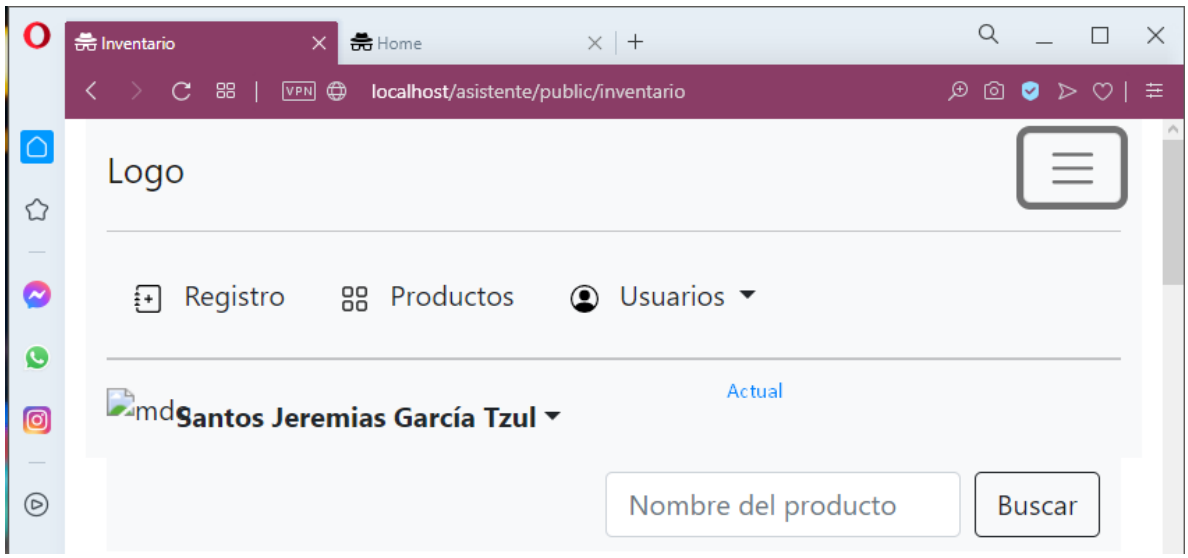


Ilustración 164. Menú nuevo



Formulario de registro: fue necesario modificar los campos que son requeridos por el usuario.

Ilustración 165. Registro de producto antes.

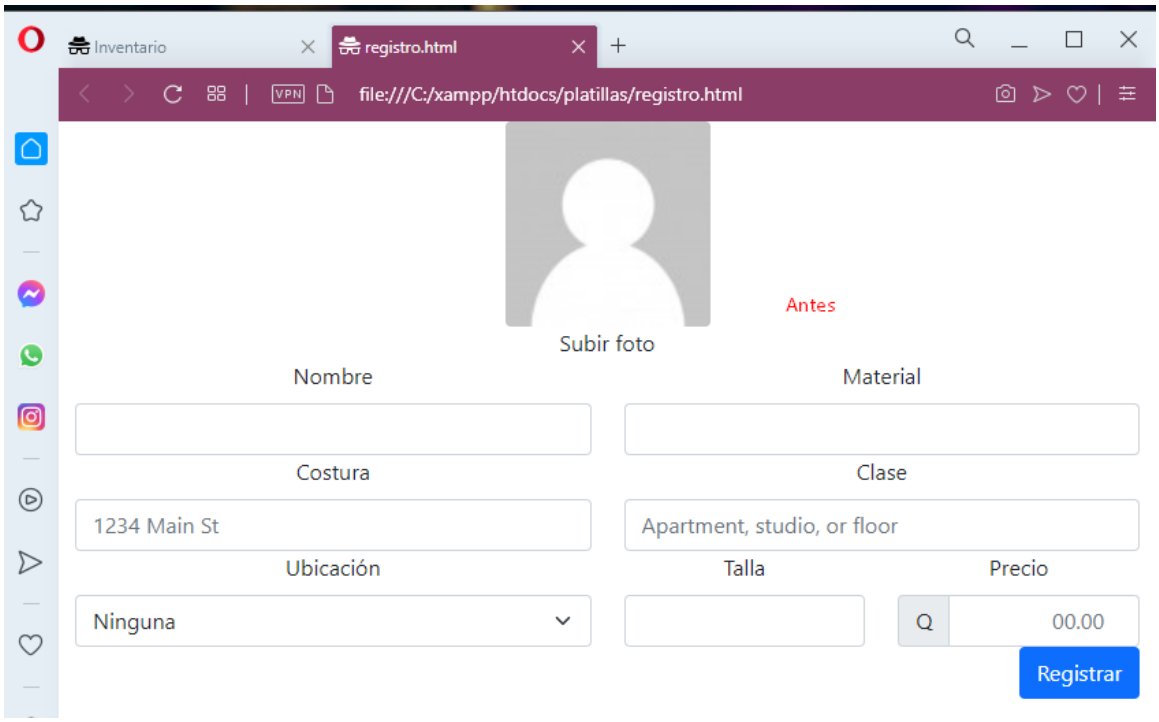
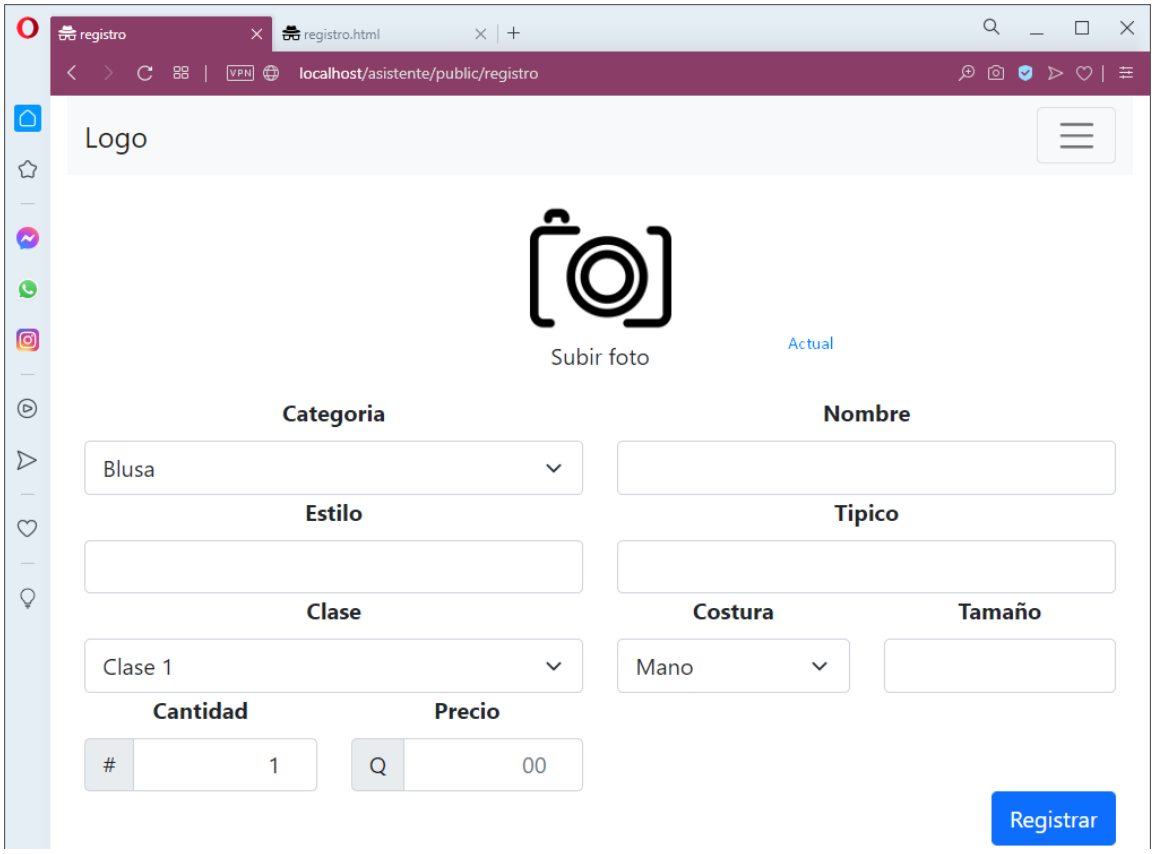


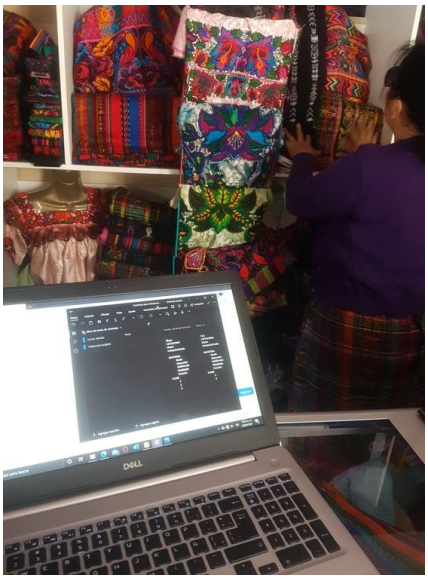
Ilustración 166. Registro de producto nuevo



3. Lógica de programación.

En la lógica de programación fue necesario realizar visitas al cliente para determinar tanto los campos que se necesitaban llenar, los datos que se presentan al momento de realizar una consulta y como de la misma manera se fue probando las funcionalidades de la referencia para determinar la opción más acertada.

Ilustración 167. Visita con el cliente



G. REFINAMIENTO DEL PROTOTIPO

Con la evolución requerida para el prototipo se fue modificando el código según las necesidades del cliente, una de las funciones con más modificación fue la del sistema de referencia ya que las funciones que se crearon no mostraban la información adecuada.

Fue necesario crear dos métodos para refinar la referencia como lo veremos a continuación:

Método 1: Se realizó una consulta para todos los productos de la base de datos, teniendo ya la consulta realizada se recorría el objeto para segmentar todas aquellas que tenían coincidencia, pero se volvía un código muy grande y dificultaba el tiempo en procesar la información.

Ilustración 168. Primer método

```
public function asistente($id){
    $query = item::select('id','color1','color2','color3','color4','color5','categoria')->find($id);
    $items = item::select('id','color1','color2','color3','color4','color5','categoria','imagen','nombre','precio')->get;
    $concidencia = [];
    foreach ($items as $item){
        if ($query->color1 = $item->color1) {
            $concidencia[$item->id] += 1;
        }
        if ($query->color1 = $item->color2) {
            $concidencia[$item->id] += 1;
        }
        if ($query->color1 = $item->color3) {
            $concidencia[$item->id] += 1;
        }
        if ($query->color1 = $item->color4) {
            $concidencia[$item->id] += 1;
        }
        if ($query->color1 = $item->color5) {
            $concidencia[$item->id] += 1;
        }
    }
    return view('cliente.asistente', compact('tquery', 'query'));
}
```

Método 2: fue necesario modificar el método presentado para mejorar el tiempo de respuesta al presentar las referencias y sobre todo que los resultados sean los más certeros posibles.

Ilustración 169. Segundo método

```
public function asistente($id){
    $query = item::select('id','color1','color2','color3','color4','color5','categoria')->find($id);
    $tquery = item::select('id','color1','color2','color3','color4','color5','categoria','imagen','nombre','precio')
        ->orWhere('color1', $query->color1)
        ->orWhere('color2', $query->color1)
        ->orWhere('color3', $query->color1)
        ->orWhere('color4', $query->color1)
        ->orWhere('color5', $query->color1)
        ->orWhere('color1', $query->color2)
        ->orWhere('color2', $query->color2)
        ->orWhere('color3', $query->color2)
        ->orWhere('color4', $query->color2)
        ->orWhere('color5', $query->color2)
        ->orWhere('color1', $query->color3)
        ->orWhere('color2', $query->color3)
        ->orWhere('color3', $query->color3)
        ->orWhere('color4', $query->color3)
        ->orWhere('color5', $query->color3)
        ->orWhere('color1', $query->color4)
        ->orWhere('color2', $query->color4)
        ->orWhere('color3', $query->color4)
        ->orWhere('color4', $query->color4)
        ->orWhere('color5', $query->color4)
        ->orWhere('color1', $query->color5)
        ->orWhere('color2', $query->color5)
        ->orWhere('color3', $query->color5)
        ->orWhere('color4', $query->color5)
        ->orWhere('color5', $query->color5)
        ->get();
    return view('cliente.asistente', compact('tquery', 'query'));
}
```

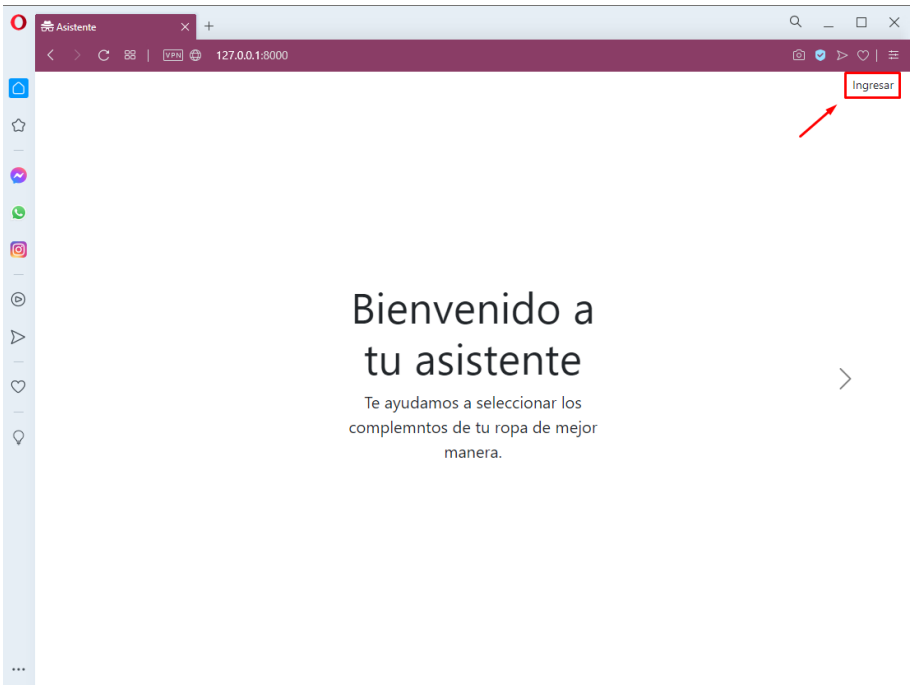
VI. RESULTADO

Con fundamento a todo lo presentado en el capítulo anterior, se presenta el resultado teniendo un prototipo funcional a los requerimientos del cliente, las principales funcionalidades como lo es el catálogo personalizado gracias a su sistema de referencia a los productos basados en los colores que estos tienen y que son presentados para ayudar al cliente en la elección de un traje.

1. Módulo de administración

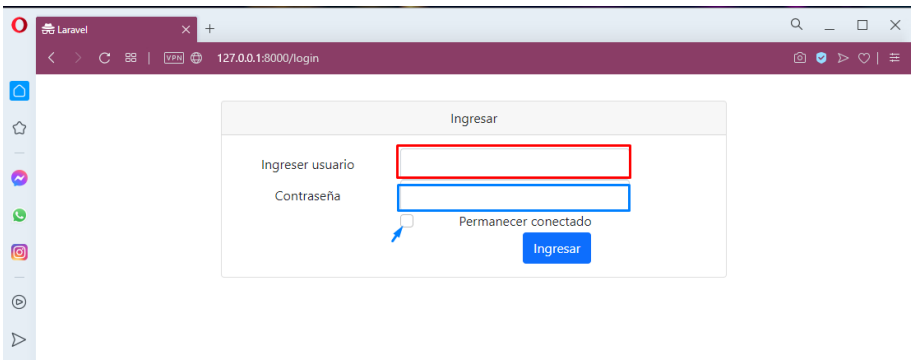
Página principal: para ingresar al módulo es necesario ir a la parte superior donde aparece la opción de ingresar que redirige al login.

Ilustración 170. Página principal



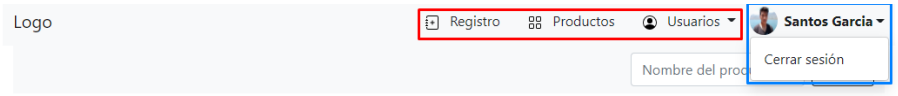
Login: ingresamos nuestras credenciales que es el **usuario** y **contraseña**, nos muestra la opción de permanecer conectado para que el inicio sea permanente hasta cerrar la sesión.

Ilustración 171. Página login



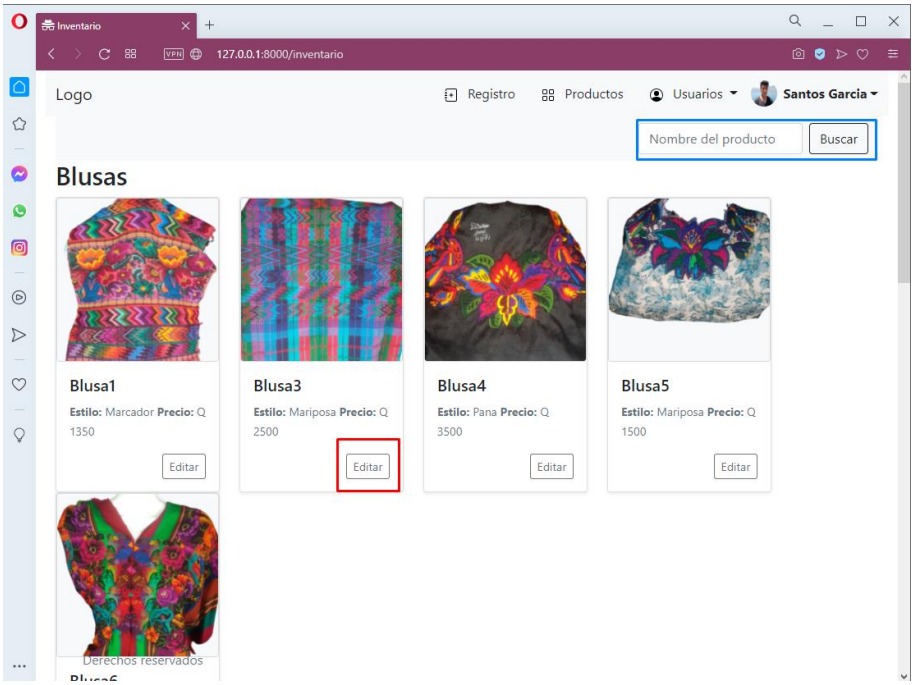
Menú de navegación: está dividida en dos, siendo la primera para ingresar a todas las rutas o páginas como lo es **registrar** un producto, revisar los **productos** y administrar los **usuarios** de ser un administrador. En la segunda parte donde podrá **cerrar sesión**.

Ilustración 172. Menú del módulo de administración



Inventario: donde podrá el usuario consultar algún producto por nombre, y desde el mismo poder actualizar los datos del producto.

Ilustración 173. Página para ver productos



Inventario-buscar: en este apartado podemos buscar un producto por el nombre identificativo como lo veremos a continuación.

Ilustración 174. Buscador de producto por nombre

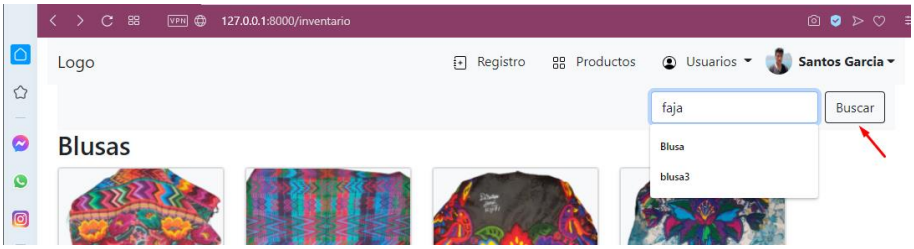
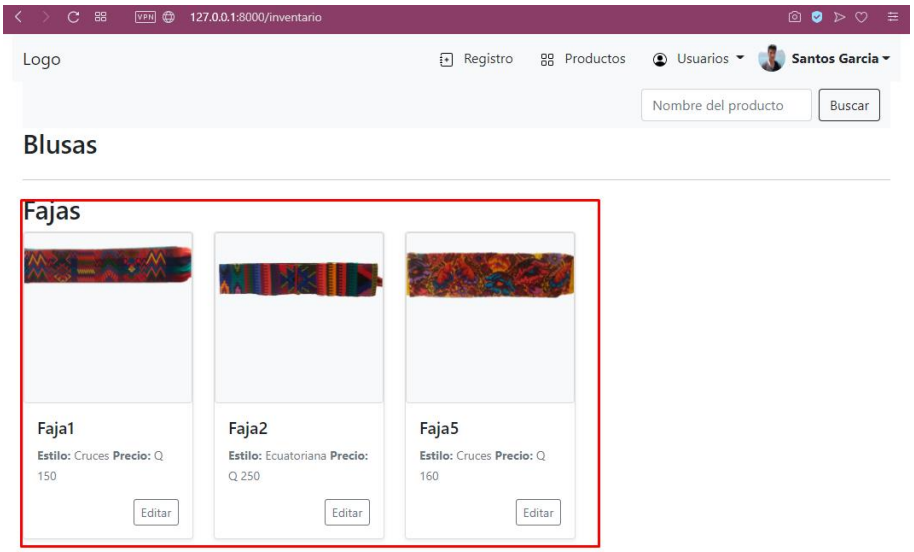
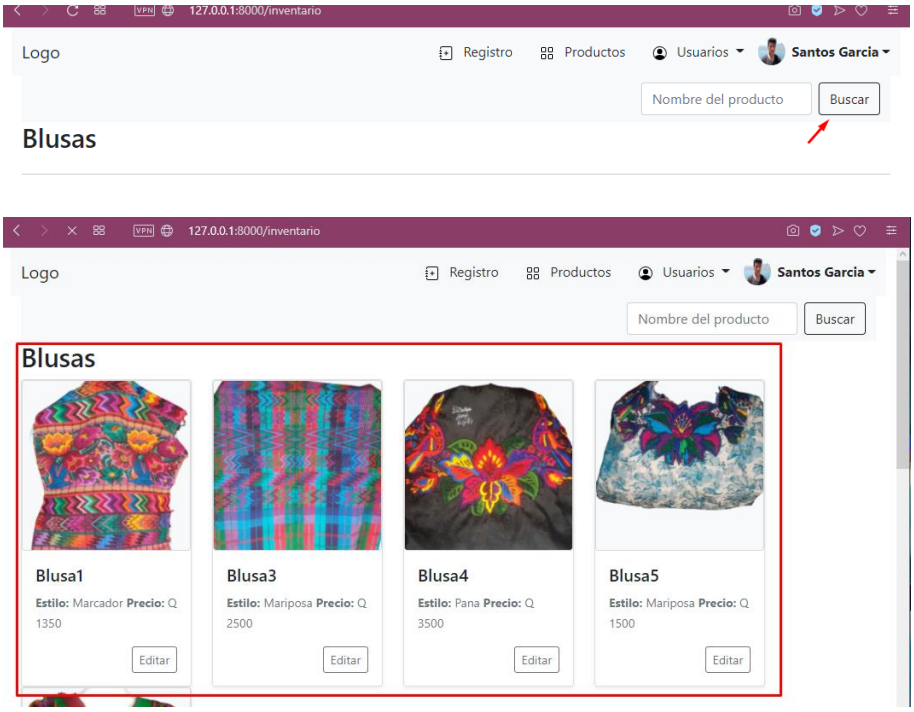


Ilustración 175. Resultado de búsqueda por nombre



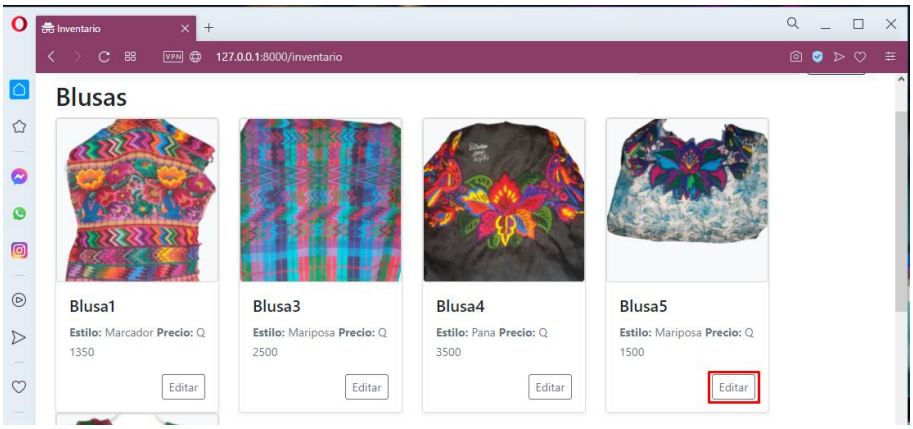
Para consultar todos los productos registrados podemos dar clic de nuevo al botón buscar.

Ilustración 176. Consultar de nuevo los productos



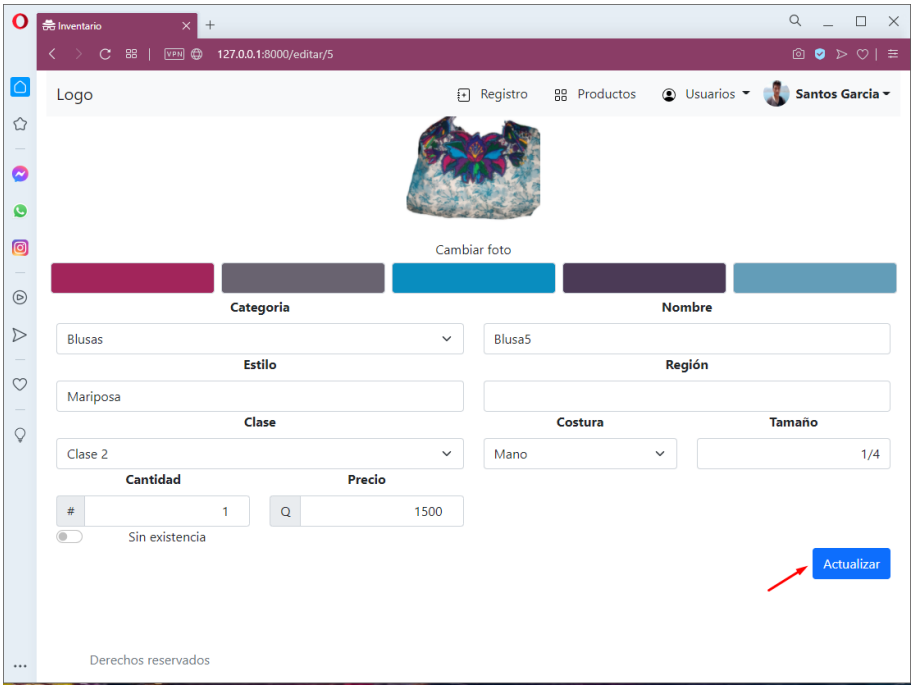
Editar un producto: damos clic en el botón **editar** sobre el producto que deseamos modificar y esto nos llevará a la página donde podemos actualizar los datos del producto.

Ilustración 177. Ingresar a editar un producto



Dentro de la página de editar podemos modificar los datos del producto y después de modificar podremos dar clic al botón Actualizar.

Ilustración 178. Página para editar los datos del producto



Registrar producto: ingresamos desde el menú a la opción de **registro** y mostrará la página donde podremos registrar los productos.

Ilustración 179. Ingresar a la página para registrar un producto

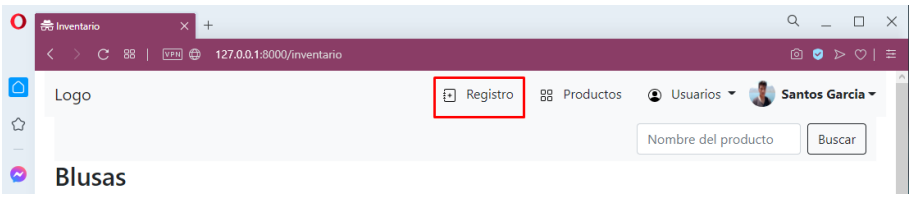
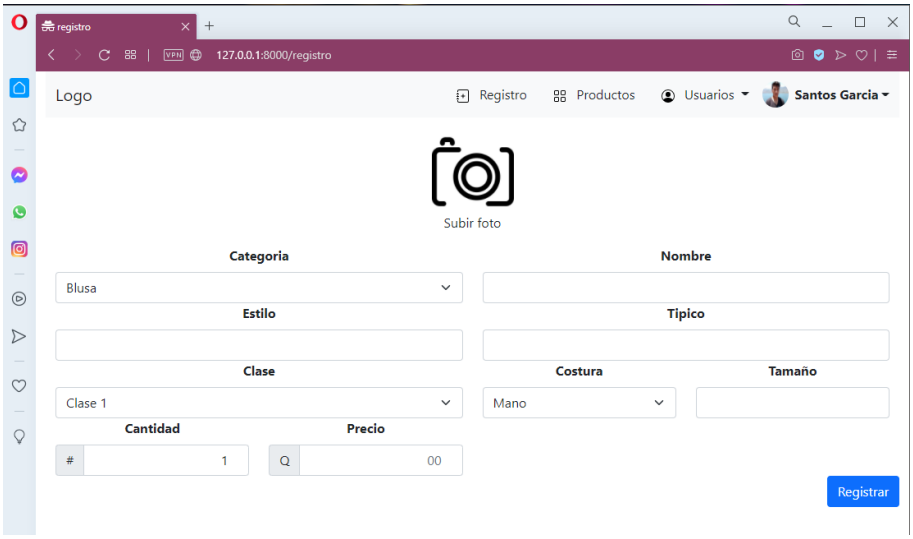
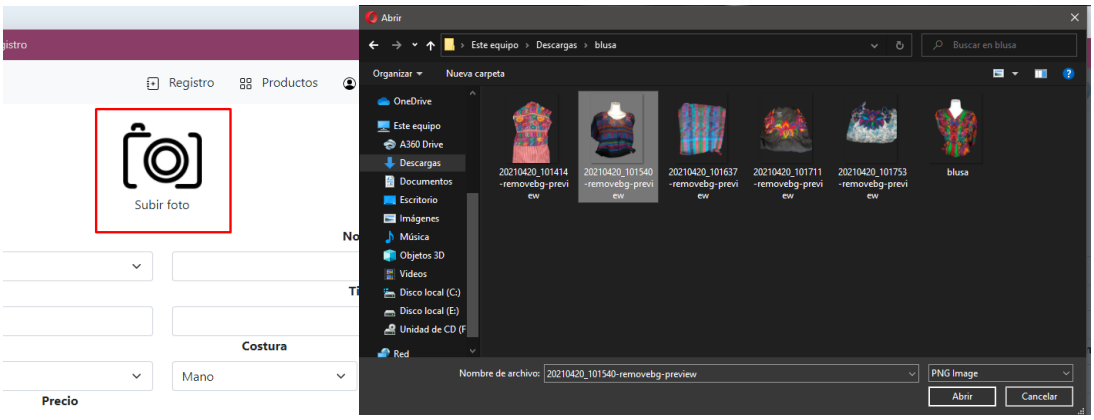


Ilustración 180. Formulario para llenar datos del producto



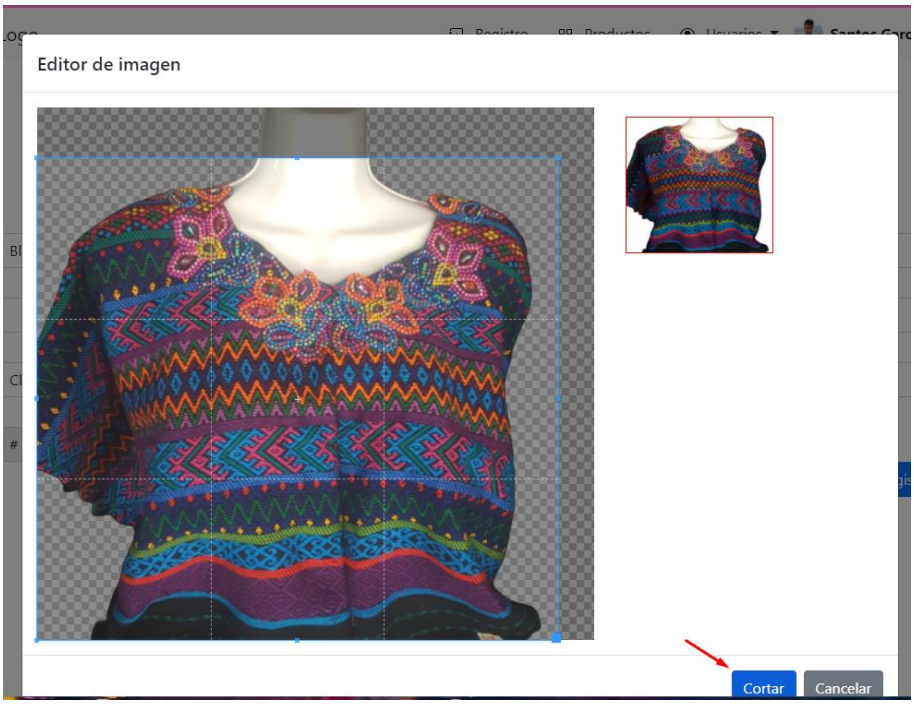
Para agregar la fotografía damos clic sobre el icono de la cámara y seleccionamos desde archivos la imagen.

Ilustración 181. Selección de imagen para registro



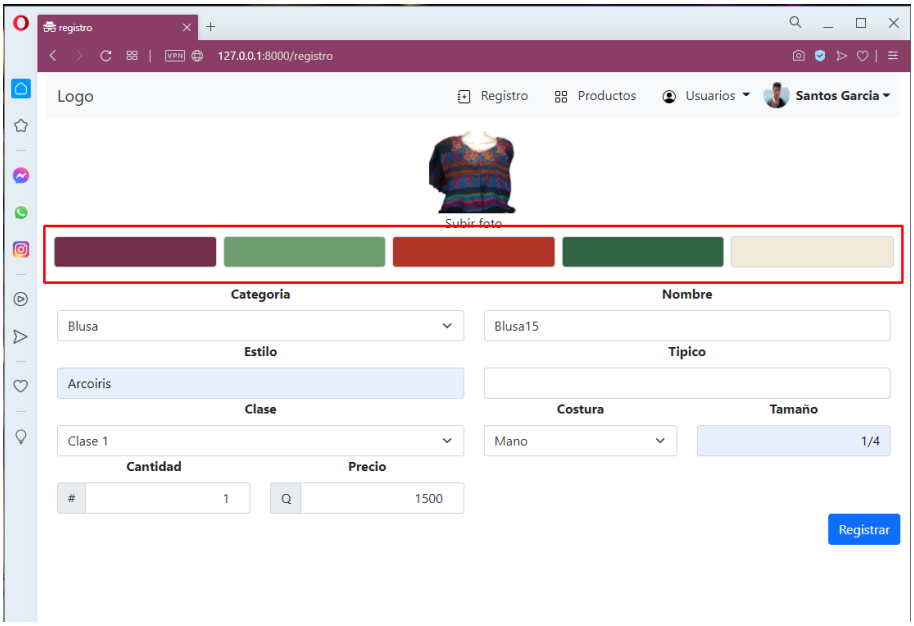
Recortamos la imagen a la medida necesaria para que solo centremos el producto, teniendo una vista previa en la parte derecha del editor de imágenes.

Ilustración 182. Editor de imágenes



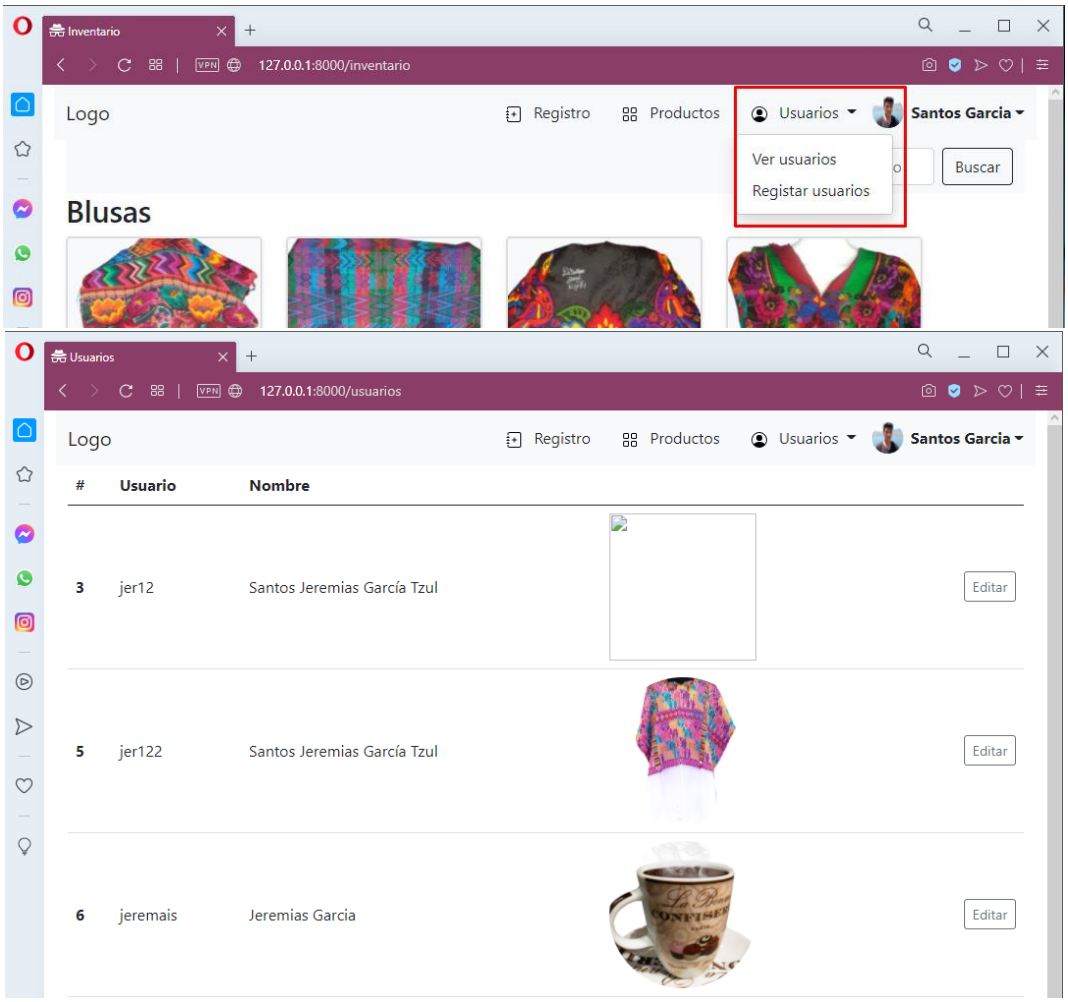
Al momento de recortar la imagen el sistema tomara los colores de muestra, luego llenamos los campos del formulario.

Ilustración 183. Sistema de toma de colores



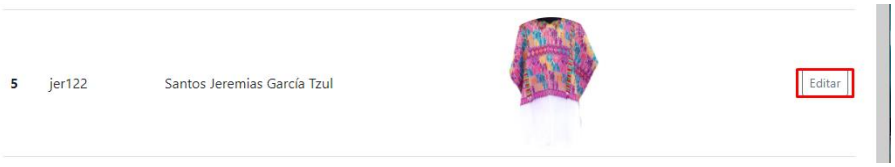
Ver usuarios: en la ventana de ver usuario encontraremos los usuarios registrados que pueden acceder al sistema concretamente al módulo de administración. Ingresamos desde el menú en la opción de usuarios.

Ilustración 184. Consultar usuario del sistema



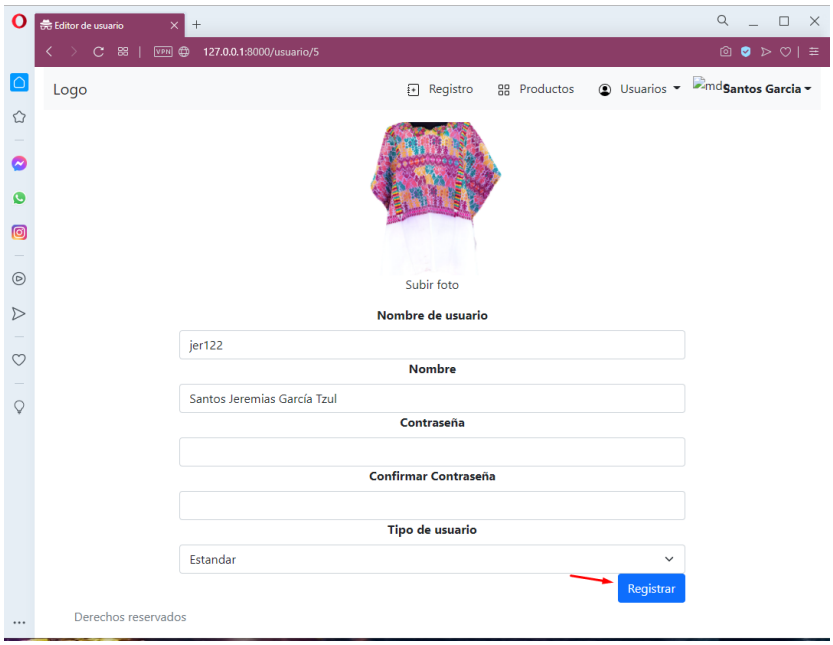
Editar usuario: dentro de la opción que se presenta podremos modificar el usuario dando clic al botón editar.

Ilustración 185. Editar usuario del sistema



Editar usuario: estando en la página de editar usuario podremos modificar los datos del usuario y damos clic al botón Actualizar.

Ilustración 186. Página para actualizar un usuario



Registrar usuario: de la misma manera ingresamos desde el menú en la sección de usuario, pero esta vez en Registrar usuario.

Ilustración 187. Ingresar a registra un usuario

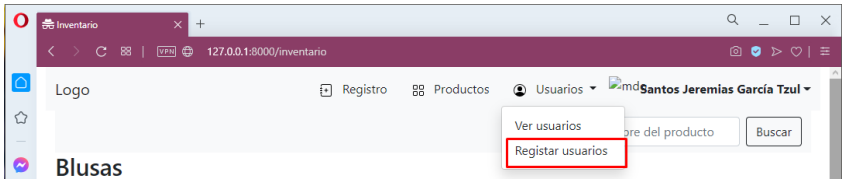
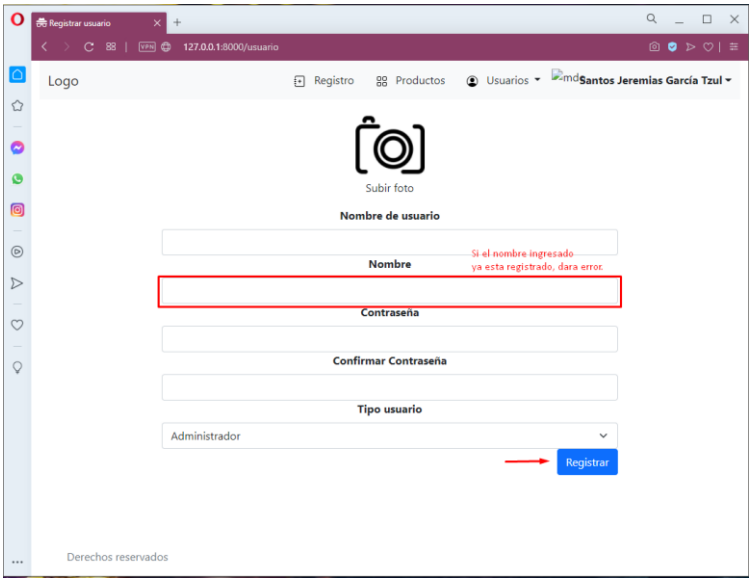


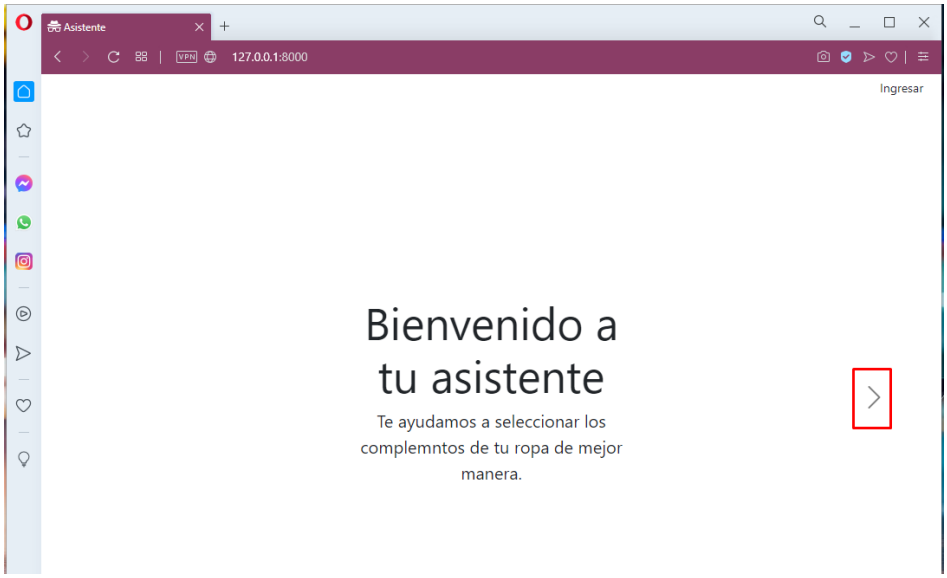
Ilustración 188. Registrar un nuevo usuario del sistema



2. Modulo cliente

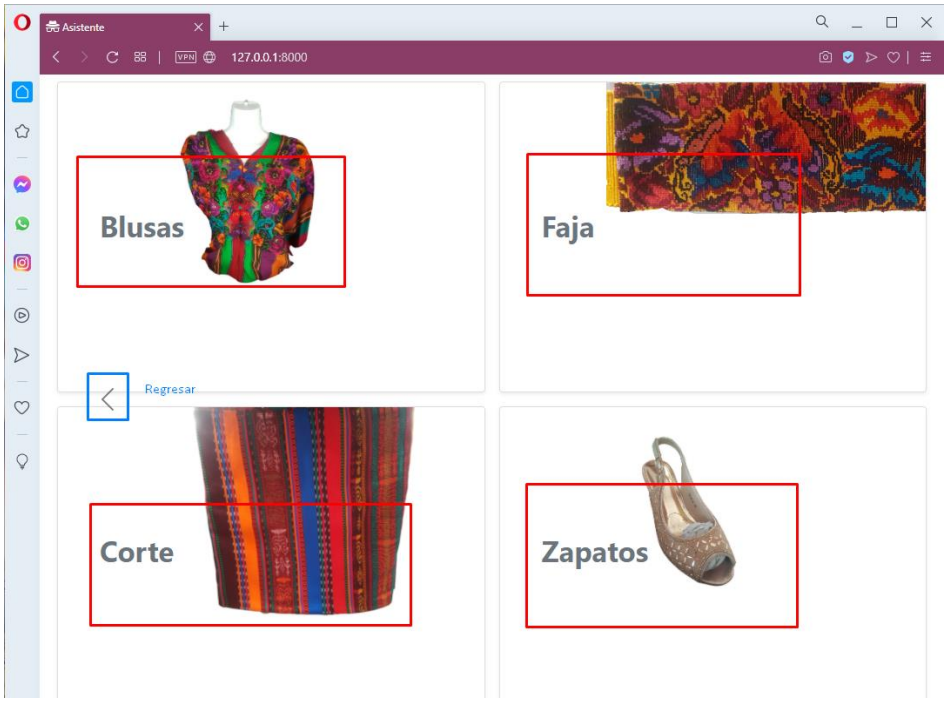
Asistente: ayuda al cliente a seleccionar el producto que busca en base a la categoría, dando al botón de siguiente.

Ilustración 189. Página de asistente



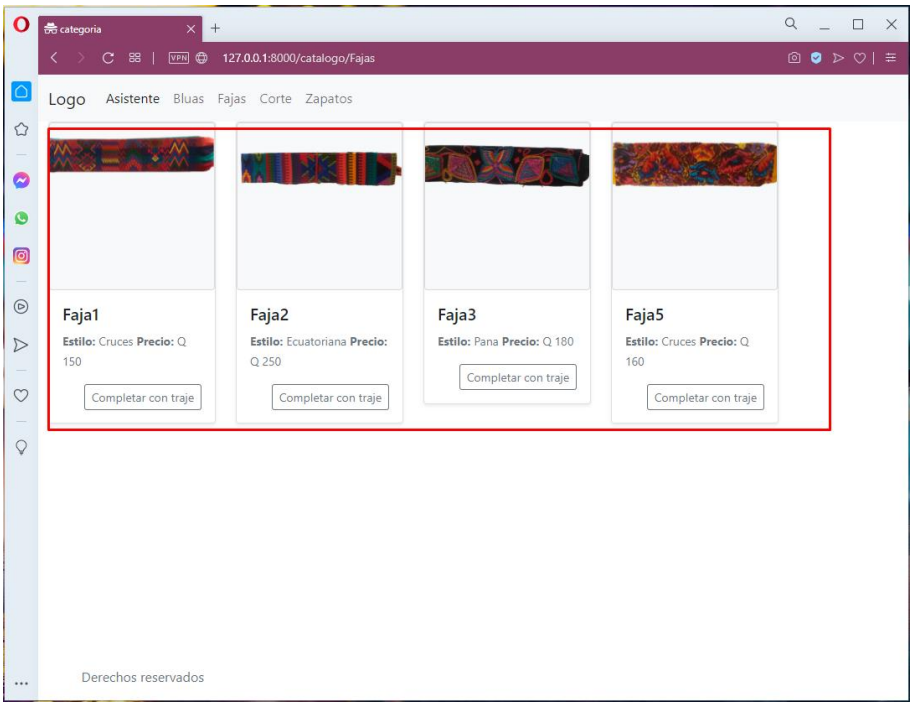
Seleccionar la categoría que busca el cliente y sí quiere regresar al menú puede presionar el botón de atrás.

Ilustración 190. Seleccionar categoría



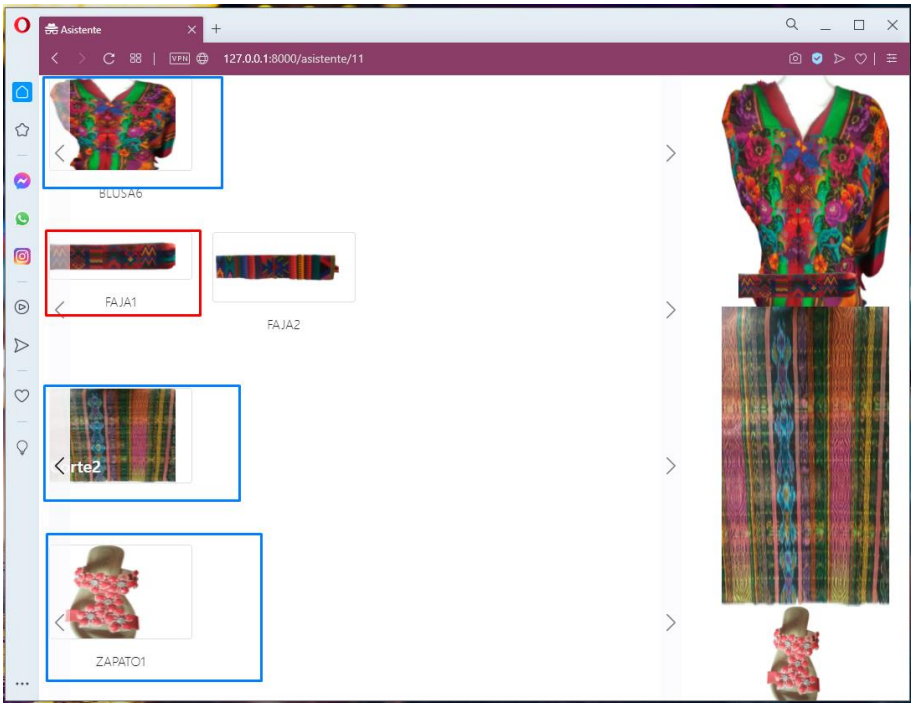
Catálogo: muestra los productos relacionados a la categoría seleccionada.

Ilustración 191. Página de catálogo



Referencia: Para ingresar a la referencia será necesario darle clic al botón **completar con traje** del producto que desea. Los remarcados de color azul son los que el sistema de referencia presenta.

Ilustración 192. Página de referencia



VII. ANÁLISIS DE RESULTADOS

Se logró completar con cada uno de los objetivos específicos, dando como resultados las fases primordiales en la ejecución del proyecto, empleando el modelo de prototipos para desarrollar y segmentar cada proceso para la codificación adecuada de cada funcionalidad esperada.

El prototipo fue aprobado por el cliente hasta la etapa de refinamiento, con las funcionalidades esperadas, de esta manera se puede dar por concluido el proceso del desarrollo del prototipo.

Ilustración 193. Firma de entendimiento de aprobación del sistema.

VIII. CONCLUSIONES

Como resultado de la investigación realizada dentro de la tienda se logró identificar los puntos importantes en la falta de visualización e identificación de los productos, como factores importantes que hace que el tiempo para la atención de los clientes sea extenso.

Después del análisis de los resultados de la investigación se definieron los requerimientos necesarios para el desarrollo del sistema web, teniendo como resultados la estructura de interfaz de usuario y la lógica de programación que se emplearon en el desarrollo. Con estos procedimientos realizados se pudo codificar los requerimientos planteados por el cliente.

La evidencia que se presentó dentro del trabajo demuestra el cumplimiento con los objetivos planteados, dando como resultado un prototipo funcional que se puede implementar dentro de la tienda y brindar las funcionalidades que tiene con la finalidad de ayudar a los clientes a encontrar el producto deseado de forma rápida.

IX. RECOMENDACIONES

Para los desarrolladores que toman el prototipo presentado, deberán mejorar el funcionamiento de la muestra de colores de la imagen, ya que de esta manera se incrementa la posibilidad de presentar más resultados en el sistema de referencia.

De ser implementado el prototipo, deberán considerar las funcionalidades presentadas como propuestas, mejoradas y refinadas, dentro del diseño de interfaz del usuario. En caso del requerimiento de hardware se pueden emplear varias herramientas que ayuden a mejorar las funcionalidades presentadas.

Para poder mejorar el sistema de referencia se recomienda la implementación de un modelo matemático que proporcione un algoritmo avanzado, que ayude en la realización de consultas masivas; ya que por magnitud de la tienda y siendo un modelo de prototipo, se optó por la metodología de consulta.

X. BIBLIOGRAFÍA

Ebratt Gómez, R., Mancilla Herrera, A., & Capacho Portilla, J. (2016). *Diseño y construcción de algoritmos.* Universidad del Norte.

Huesos Ibáñex, L. (2015). *Administración de Sistemas Gestores de Base de Datos. 2ª Edición.* Madrid: Grupo Editorial RA-MA.

J. Date, C. (2001). *Introducción a los sistemas de bases de datos.* Pearson Educación.

Moreno Pérez, J. C. (2012). *Sistemas informáticos y redes locales.* RA-MA.

Morera Pascual, J. M., & Pérez-Campanero Atanasio, J. A. (2002). *Conceptos de sistemas operativos.* España: Univ Pontifica Comillas.

Niño Camazón, J. (2011). *Sistemas operativos monopuesto.* Editex.

Riera García, J., & Alabau Muñoz, A. (1992). *Teleinformática y redes de computadores.* Barcelona: Marcombo.

Rivera Darín, J. (2016). *Fundamentos de Redes Informáticas: 2ª Edición.* IT Campus Academy.

XI. ANEXO

Metodología, recopilación de información

Ilustración 194. Recopilación de información

Metodología, desarrollo de lógica de programación.

Ilustración 195. Instrucciones fundamentales para Laravel

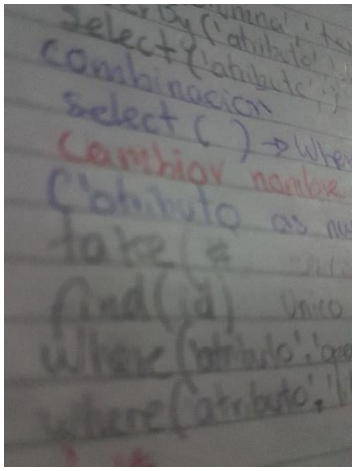


Ilustración 196. Validando cantidad de caracteres que da una conversión base 64 de un imagen

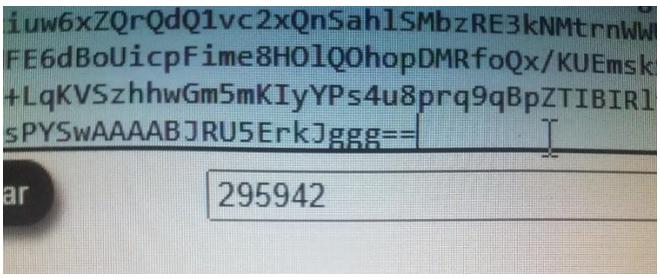


Ilustración 197. Documentación del sistema

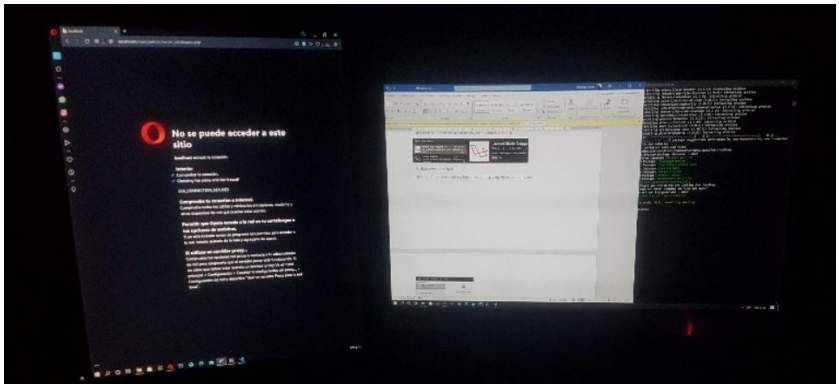
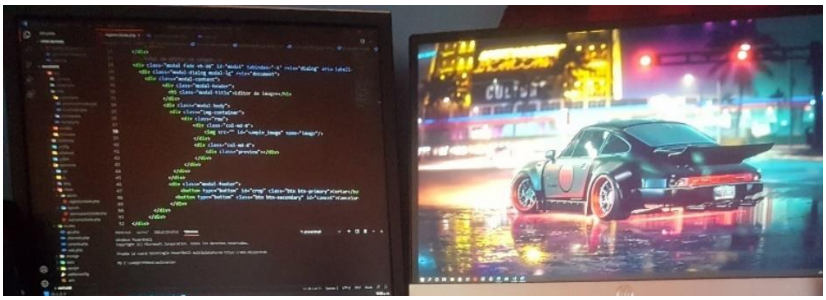


Ilustración 198. Prueba de estrés al sistema



Metodología, refinamiento del sistema.

Ilustración 199. Muestra de producto faja

Ilustración 200. Muestra de producto blusa

Ilustración 201. Muestra de producto zapato

Aprobación del sistema.

Ilustración 202. Carta de entendimiento de aprobación del sistema.

La Esperanza, Totonicapán, Totonicapán 7 de mayo de 2021

María Mazariegos
Propietaria
Tienda de ropa típica
Aldea la Esperanza Totonicapán, Totonicapán.
Presente

Estimada María

Por este medio hago la entrega del sistema: **"Sistema web RESPONSIVE de catálogo personalizado de mercadería existente como herramienta de ventas de una tienda de ropas típicas con PHP y MySQL"**. A su vez solicito amablemente la aprobación del mismo, desarrollado por el estudiante Santos Jeremías García Tzul con Carné No. 151057, de la Universidad del Valle de Guatemala. Requisito para que el estudiante pueda optar al grado de Licenciado en Tecnología de Sistemas Informáticos (Campus Altiplano).

Al agradecer su atención a la presente, me suscribo.

Atentamente,

Santos Jeremías García Tzul

Aprobado

XII. GLOSARIO

Aero Glass: es un efecto de tipo cristal para las ventanas del sistema operativo Windows 7, 5

Asistencia remota: ayuda que se presta a traves de internet., 5

Carousel: es un formato donde se pueden agregar información, 39

Cortana: es un asistente de virtual caracteristica de Windows 10, 5

DOS: sistema operativo de disco., 4

Editores de texto: son softwares que permite crear y modificar archivos, 7

extensión: identifica qué aplicacion está desarollada., 35

Gimp: software para editar imagens, 7

GitHub: sistema de controlador de versiones, 35

Hardware: es cun elemento físico de una computadora, 4

intérprete: traduce una instrucción, 17

IU: User Interface Es el interfaz de usuario., 4

Navegador: software que permite a los usuarios navegar en internet, 34

Periféricos: es un componente de una computadora., 4

PERL: lengueje de progrmación, 38

Persistente: sigue durando o permanece constante por un tiempo, **12**

Píxel: es una unidad básica de un imagen, 35

Procesos: operaciones que se realizan dentro de la computadora., 4

RGB: es un modelo conformado por los colores rijo, verde y azul, 35

Sintaxis: conjunto de regla que debe seuirse al escribir un código, 33

Software: conjunto de instrucciones dados a una computadora pasa solucionar un problema., 4

Subordinado: sometida a las dependencias de otro, 14

Subprocesos: es una parte de un proceso, 17

Usuario: una persona que utiliza un servicio, 4

Windows Movie Maker

Windows Movie Maker

un software de equición de videos, 5