MANUAL TECNICO WEB

1. ALCANCE:

A manera general se espera que el sistema permite a la ferretería Ferremateriales el Maestro ubicada en campeche (dirección) gestionar de forma digital sus operaciones clave. Por un lado, los visitantes y clientes pueden podrán interactuar directamente con la ferretería para consultar productos existentes en la ferretería sin necesidad de tener que desplazarse hasta allá, realizar cotizaciones y gestionar pedidos, como también, contactar al administrador de la ferretería para hacer consultas o presentar una atención al cliente ya sea por correo como también por un chat d WhatsApp. Y por otro lado como la idea es que la ferretería pueda digitalizar todo el administrador puede acceder a un panel llamado la dashboard con herramientas de gestión como ver inventario de productos, ver ventas hechas digitalmente como también ingresar las hechas en el punto físico, ingresar los pedidos que lleguen a la tienda de los proveedores y a su vez actualice el inventario, pedir soporte técnico, ver una vista con los clientes que ya han hecho compra y están registrados en el aplicativo web como también ver graficas que lleven todo el proceso de clientes, ventas filtradas por día, mes, etc.

Módulos funcionales:

- Panel del usuario con vistas de:
    - Visualización pública de productos filtrados por su categoría o nombre de producto.
    - Carrusel de promociones.
    - Soporte o ayuda con la ferretería.
    - Carrito de compra.
    - Métodos de pago o compras en línea.
    - Vista de favoritos.

- Panel de administración con vistas de:
    - Inicio (gráficas)
    - Inventario
    - Ventas (tanto en línea como físicas)
    - Proveedores
    - Clientes
    - Soporte técnico con los programadores.

2. Descripción de los procesos

- Gestión de productos:

  - Registrar nuevos productos.
  - Modificar stock.
  - Visualización de productos (con la posibilidad de buscar alguno en específico).
  - Consultar catálogo en página pública.

- Registro de ventas:

  - Registrar productos vendidos.
  - Calcular ingreso generado.

- Administración de proveedores:

  - Crear proveedores con su respectivo producto.
  - Asociar cantidad adquirida.
  - Consultar historial de compras por proveedor.

- Administración de clientes:

  - Registrar clientes.
  - Consultar historial de compras.
  - Consultar clientes existentes.

- Visualización de estadísticas:

  - Gráficas de ventas, ingresos y productos más vendidos.

- Gestión de usuario:

  - Inicio de sesión de administrador y usuario.

  - Control de acceso al dashboard para administrador de la ferretería y opciones generales para usuarios normales.

- Atención al cliente (página pública):

  - Envío de mensajes a través del formulario de contacto con el administrador de la ferretería mediante correo y WhatsApp.

- Atención al administrador:

  - Envío de formulario a través del formulario ayuda para poder contactarse con el servicio técnico de los desarrolladores.

Requerimientos funcionales:

- El sistema debe permitir registrar pedidos de los proveedores y asociarlos a productos del inventario y/o creado actualizando su stock para que luego los clientes puedan comprarlos.

- Los visitantes podrán enviar mensajes mediante un formulario público para hacer consultas.

- Los visitantes podrán hablar con el encargado de la ferretería vía WhatsApp si lo desea o considera mejor.

- Los visitantes podrán ver productos sin registrarse.

- El sistema debe mantener un registro de todas las ventas con sus detalles.

- El sistema debe sumar automáticamente el stock de productos repetidos que lleguen de los proveedores existentes.

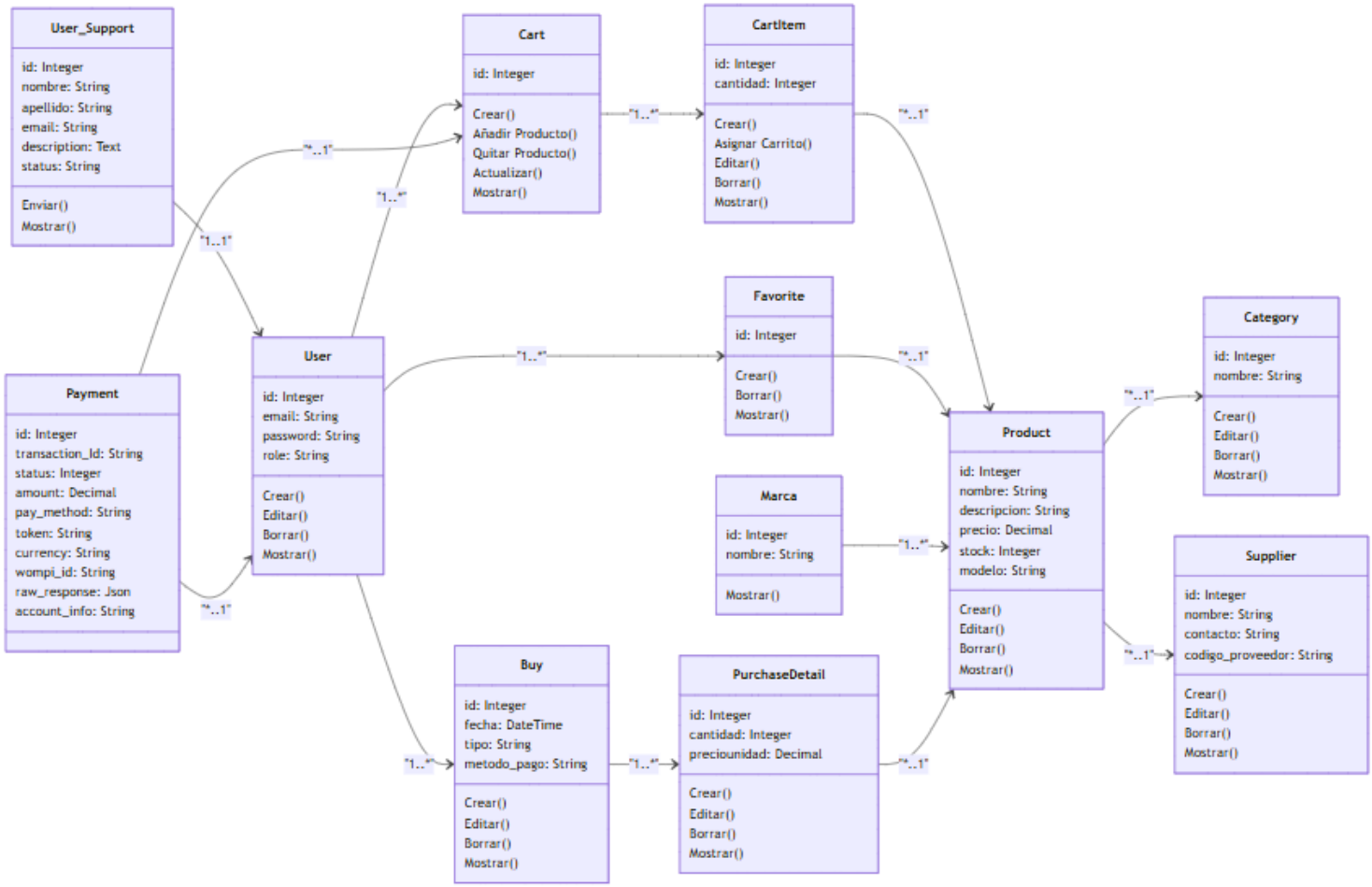
- El sistema debe proveer un dashboard con estadísticas visuales.

Requerimientos no funcionales:

- El sistema debe restringir el acceso al panel de administración mediante autenticación.
- Las contraseñas deben almacenarse de forma cifrada.
- La interfaz debe ser intuitiva y de fácil navegación para usuarios sin experiencia técnica.
- El sistema debe contar con mensajes de error claros y comprensibles.
- El sistema debe ser capaz de responder a las consultas de productos rápido.
- El dashboard debe cargar los gráficos ágilmente y en poco tiempo aún con hasta 10,000 registros (falta mejoras).
- El sistema debe ser accesible desde navegadores modernos (Chrome, Firefox, Edge).
- El sistema debe ser adaptable a dispositivos móviles y tablets (diseño responsive).

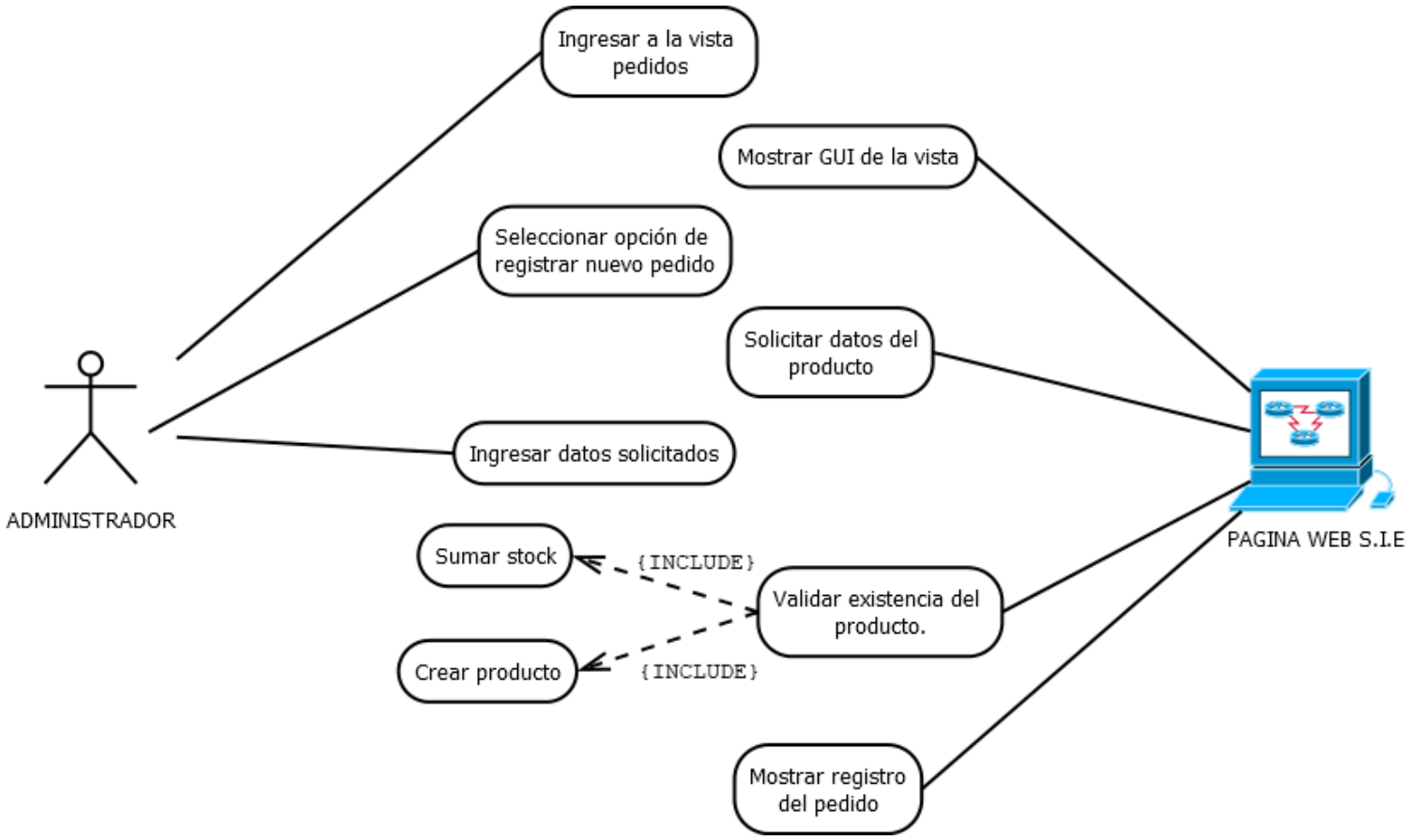
3. Diagramas UML

DIAGRAMA DE CLASE



R01

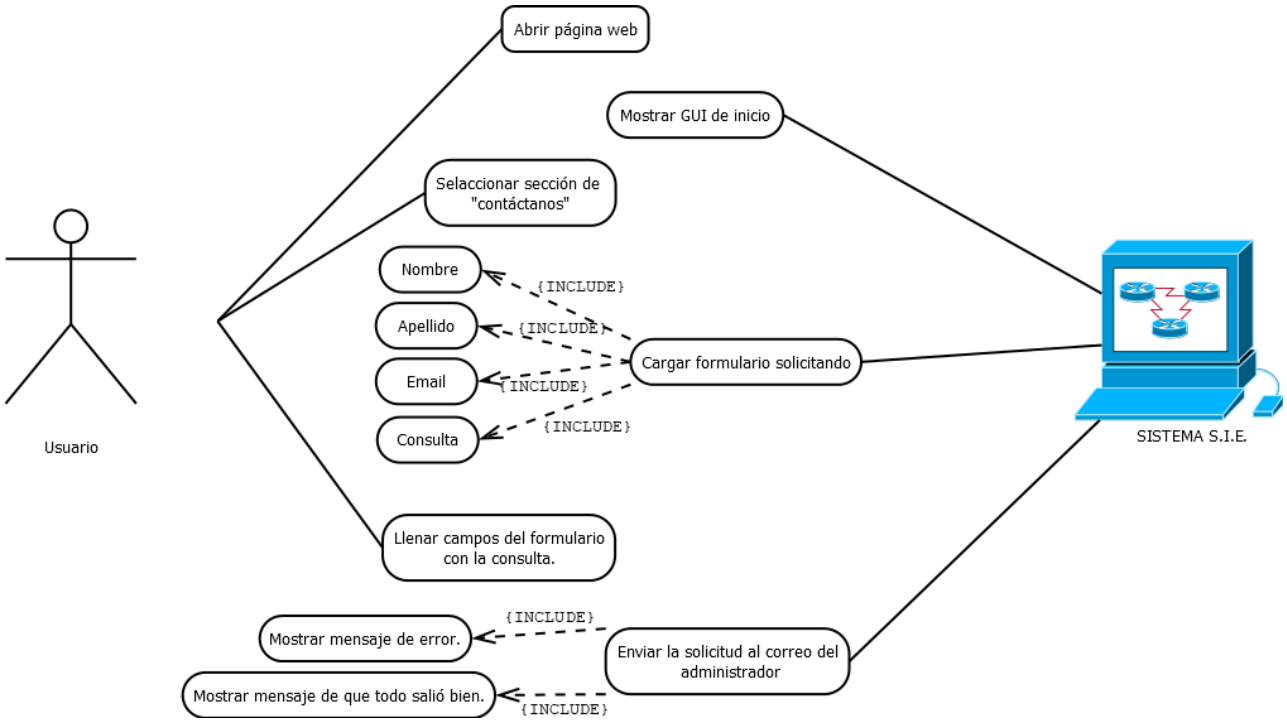
NOMBRE: REGISTRAR PEDIDO DE PROVEEDOR CON ACTUALIZACIÓN DE STOCK



FECHA: 16/09/25		
CASO DE USO 01		
NOMBRE DEL CASO DE USO: REGISTRAR PRODUCTOS CON ACTUALIZACIÓN DE STOCK		
AUTOR: KEINER LINDARTE.		
ACTOR: USUARIO Y SISTEMA		
PRECONDICION: EL ADMINISTRADOR DEBE TENER UN DISPOSITIVO CON CONEXIÓN A INTERNET DICHO DISPOSITIVO		
TAMBIEN DEBE TENER ACCESO A LA PÁGINA, EL ADMINISTRADOR DEBE YA TENER LA SECCIÓN INICIADA CON SU DEBIDO ROL.		
FLUJO NORMAL	PASOS	DESCRIPCIONES
	1	EL ADMINISTRADOR INGRESA A LA VISTA DE PEDIDOS
	2	EL SISTEMA MUESTRA LA GUI DE LA VISTA SELECCIONADA.
	3	EL ADMINISTRADOR SELECCIONA LA OPCIÓN PARA REGISTRAR UN NUEVO PEDIDO QUE LLEGÓ DE UN PROVEEDOR
	4	EL SISTEMA SOLICITA LOS DATOS DEL PRODUCTO RECIBIDO.
	5	EL ADMINISTRADOR INGRESA LOS DATOS SOLICITADOS
	6	EL SISTEMA VERIFICA SI EL PRODUCTO YA EXISTE EN LA TABLA PRODUCTS SI ES ASÍ, EL SISTEMA SUMA LA CANTIDAD
		AL STOCK ACTUAL Y SI NO EXISTE EL SISTEMA CREA EL NUEVO PRODUCTO CON SU STOCK INICIAL.
	7	SI TODO SALIÓ BIEN EL SISTEMA MOSTRARÁ EL NUEVO REGISTRO.
FLUJO ALTERNA	PASOS	DESCRIPCIONES
	2.1	EL SISTEMA PRESENTA FALLA Y NO MUESTRA LA VISTA DE PEDIDOS
	4.1	EL SISTEMA PRESENTA FALLA Y NO ARROJA EL FORMULARIO PARA INGRESAR DATOS DEL PRODUCTO
	6.1	EL SISTEMA NO LOGRA REALIZAR LA CONSULTA A LA BASE DE DATOS
	6.2	EL SISTEMA NO ACTUALIZA STOCK.
	6.3	EL SISTEMA EN CASO DE NO EXISTIR EL PRODUCTO NO LO CREA CON SU STOCK INICIAL
	7.1	EL SISTEMA NO MUESTRA EL NUEVO REGISTRO EN EL HISTORIAL
POSTCONDICION: EL REGISTRO DEL PEDIDO SE MOSTRARÁ EN EL HISTORIA Y EL INVENTARIO SE ACTUALIZARÁ CON EL NUEVO STOCK Y/O PRODUCTO		

R02

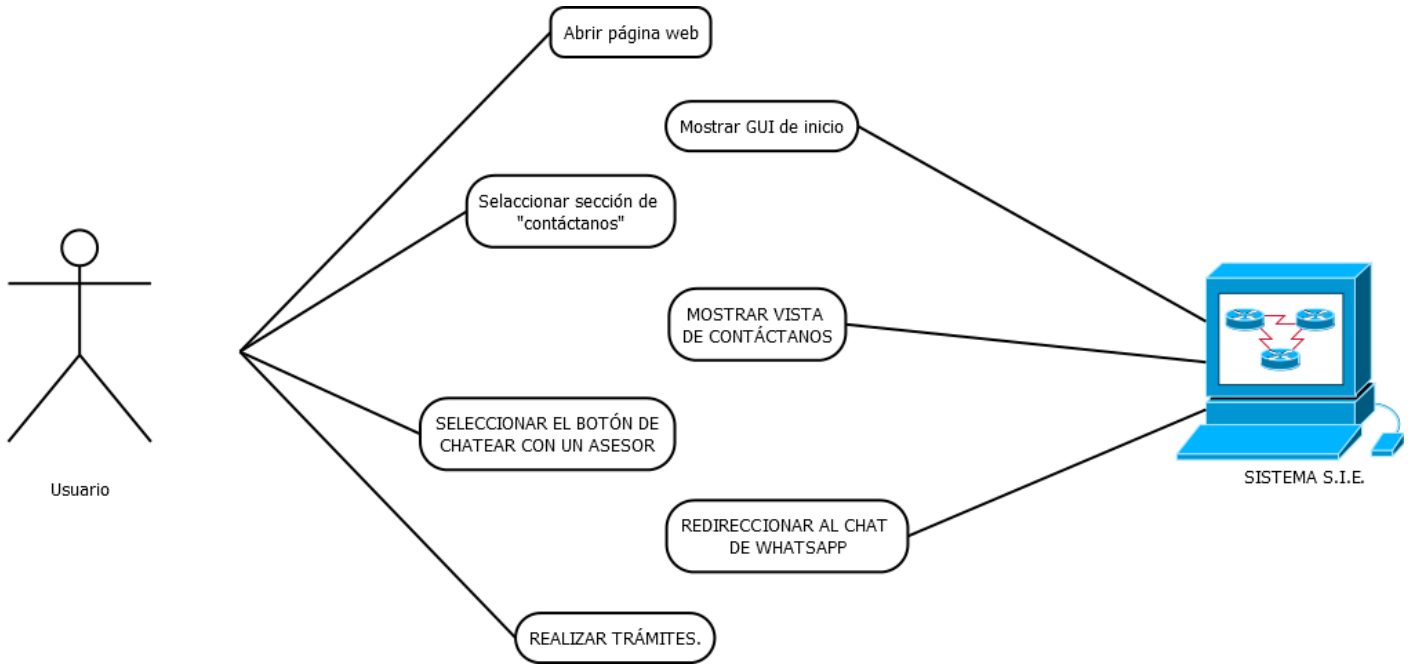
NOMBRE: FORMUALRIO DE CONSULTAS.



FECHA: 16/09/25		
CASO DE USO 02		
NOMBRE DEL CASO DE USO: FORMUALRIO DE CONSULTAS.		
AUTOR: KEINER LINDARTE.		
ACTOR: USUARIO Y SISTEMA		
PRECONDICION: EL USUARIO DEBE TENER UN DISPOSITIVO CON CONEXIÓN A INTERNET DICHO DISPOSITIVOTAMBIEN DEBE TENER ACCESO A LA PÀGINA		
FLUJO NORMAL	PASOS	DESCRIPCIONES
	1	EL USUARIO INGRESA A LA PÁGINA SIE DESDE SU NAVEGADOR DE CONFIANZA
	2	EL SISTEMA LE MUESTRA LA GUI DE INICIO DEL APLICATIVO WEB
	3	EL USUARIO ACCEDE A LA VISTA "CONTÁCTANOS".
	4	EL SISTEMA MUESTRA EL FORMULARIO SOLICITANDO NOMBRE, APELLIDO, CORREO Y MENSAJE, TAMBIEN MUESTRA UN BOTÓN PARA COMUNICARSE MEDIANTE WHATSAPP .
	5	EL USUARIO SELECCIONA EL BOTÓN DE CHATEAR CON UN ASESOR
	6	EL SISTEMA ENVIA DICHA CONSULTA AL CORREO DEL ADMINISTRADOR Y A SU VEZ ARROJA EL MENSAJE YA SEA DE ERROR O DE QUE TODO SE ENVIÓ CORRECTAMENTE Y PRO
		EL ADMINISTRADOR DE LA FERRETERÍA SE COMUNICARÁ CON EL USUARIO.
FLUJO ALTERNA	PASOS	DESCRIPCIONES
	2.1	EL SISTEMA PRESENTA FALLA Y NO MUESTRA EL INICIO DE LA PÁGINA
	4.1	EL SISTEMA PRESENTA FALLA Y NO MUESTRA LA VISTA DE "CONTÁCTANOS"
	6.1	EL SISTEMA NO LOGRA PROCESAR Y ENVIAR LA SOLICITUD AL CORREO DEL ADMINISTRADOR
	6.2	EL SISTEMA NO DEVUELVE NINGÚN AVISO.
POSTCONDICIÓN: EL USUARIO LOGRA HACER SU CONSULTA Y EL ADMINISTRADOR LE RESPONDE AL CORREO		

R03

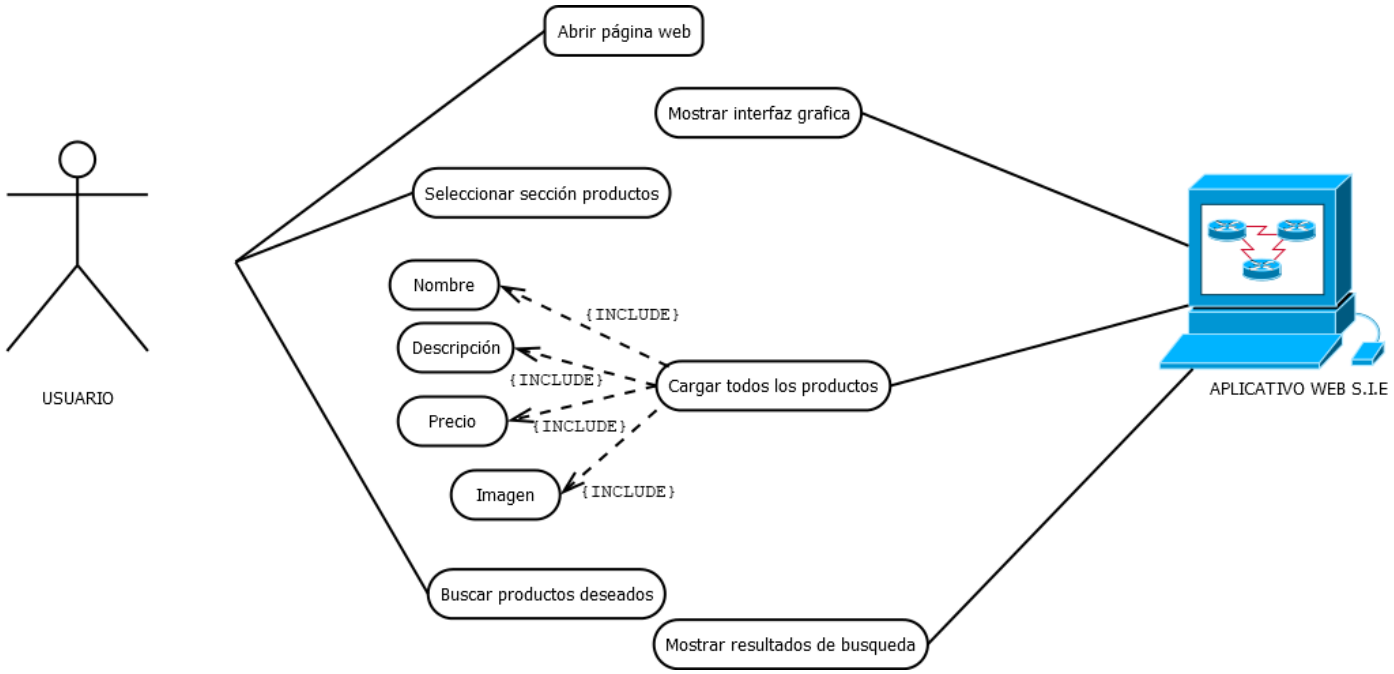
NOMBRE: ATENCIÓN AL CLIENTE VÍA WHATSAPP.



FECHA: 16/09/25		
CASO DE USO 03		
NOMBRE DEL CASO DE USO: ATENCIÓN AL CLIENTE VÍA WHATSAPP		
AUTOR: KEINER LINDARTE.		
ACTOR: USUARIO Y SISTEMA		
PRECONDICION: EL USUARIO DEBE TENER UN DISPOSITIVO CON CONEXIÓN A INTERNET DICHO DISPOSITIVOTAMBIEN DEBE TENER ACCESO A LA PÀGINA		
FLUJO NORMAL	PASOS	DESCRIPCIONES
	1	EL USUARIO INGRESA A LA PÁGINA SIE DESDE SU NAVEGADOR DE CONFIANZA.
	2	EL SISTEMA LE MUESTRA LA GUI DE INICIO DEL APLICATIVO WEB.
	3	EL USUARIO ACCEDE A LA VISTA "CONTACTANOS".
	4	EL SISTEMA MUESTRA EL FORMULARIO SOLICITANDO NOMBRE, APELLIDO, CORREO Y MENSAJE, TAMBIEN MUESTRA UN BOTÓN PARA COMUNICARSE MEDIANTE WHATSAPP .
	5	EL USUARIO COMPLETA LOS CAMPOS DEL FORMULARIO Y PRESIONA EL BOTÓN DE ENVIAR.
	6	EL SISTEMA REDIRIGE AL USUARIO AL CHAT DE WHATSAPP DE LA FERRETERIA.
	7	EL USUARIO HACE SU CONSULTA Y HABLA CON UN ASESOR.
FLUJO ALTERNA	PASOS	DESCRIPCIONES
	2.1	EL SISTEMA PRESENTA FALLA Y NO MUESTRA EL INICIO DE LA PÀGINA.
	4.1	EL SISTEMA PRESENTA FALLA Y NO MUESTRA LA VISTA DE "CONTÁCTANOS"
	6.1	EL SISTEMA PRESENTA FALLA Y NO REDIRIGE AL USUARIO AL CHAT ESTABLECIDO.
POSTCONDICIÓN: EL CLIENTE LOGRA CONTACTAR AL ASESOR MEDIANTE EL CHAT DE WHATSAPP Y RESOLVER SUS DUDAS.		

R04

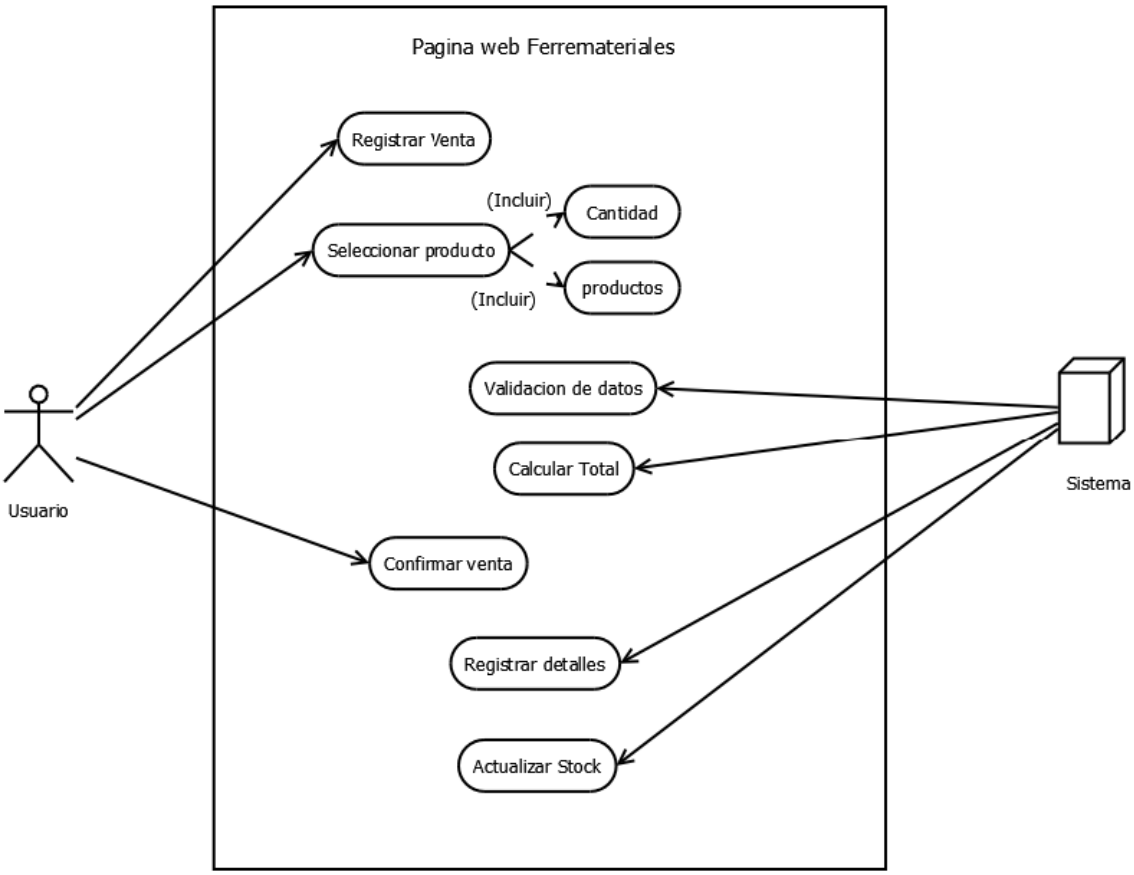
NOMBRE: VER PRODUCTOS SIN REGISTRARSE



FECHA: 16/09/25		
CASO DE USO 04		
NOMBRE DEL CASO DE USO: VER PRODUCTOS SIN REGISTRARSE		
AUTOR: KEINER LINDARTE.		
ACTOR: USUARIO Y SISTEMA		
PRECONDICION: EL USUARIO DEBE TENER UN DISPOSITIVO CON CONEXIÓN A INTERNET DICHO DISPOSITIVOTAMBIEN DEBE TENER ACCESO A LA PÁGINA		
FLUJO NORMAL	PASOS	DESCRIPCIONES
	1	EL USUARIO INGRESA A LA PÁGINA SIE DESDE SU NAVEGADOR DE CONFIANZA.
	2	EL SISTEMA MUESTRA PÁGINA DE INICIO
	3	EL USUARIO SELECCIONA LA OPCIÓN DE VER PRODUCTOS
	4	EL SISTEMA LES MUESTRA TODOS LOS PRODUCTOS EN STOCK DE LA FERRETERIA CON SU NOMBRE, DESCRIPCIÓN, PRECIO, IMAGEN.
	5	EL USUARIO BUSCA EL PRODUCTO DESEADO YA SEA POR NOMBRE, CATEGORIA.
	6	EL SISTEMA ARROJA RESULTADOS DE LA BUSQUEDA.
FLUJO ALTERNA	PASOS	DESCRIPCIONES
	2.1	EL SISTEMA PRESENTA FALLA Y NO MUESTRA LA PÁGINA DE INICIO.
	4.1	EL SISTEMA NO CARGA PRODUCTOS
	6.1	EL SISTEMA NO ARROJA RESULTADOS DE BUSQUEDA.
POSTCONDICIÓN: EL USUARIO PUEDE OBSERVAR LOS PRODUCTOS FILTRADOS O NO PARA ENCONTRAR EL QUE SE ADAPTE A SUS NECESIDADES Y GUSTOS SIN NECESIDAD DE CREAR E INICIAR SECCIÓN		

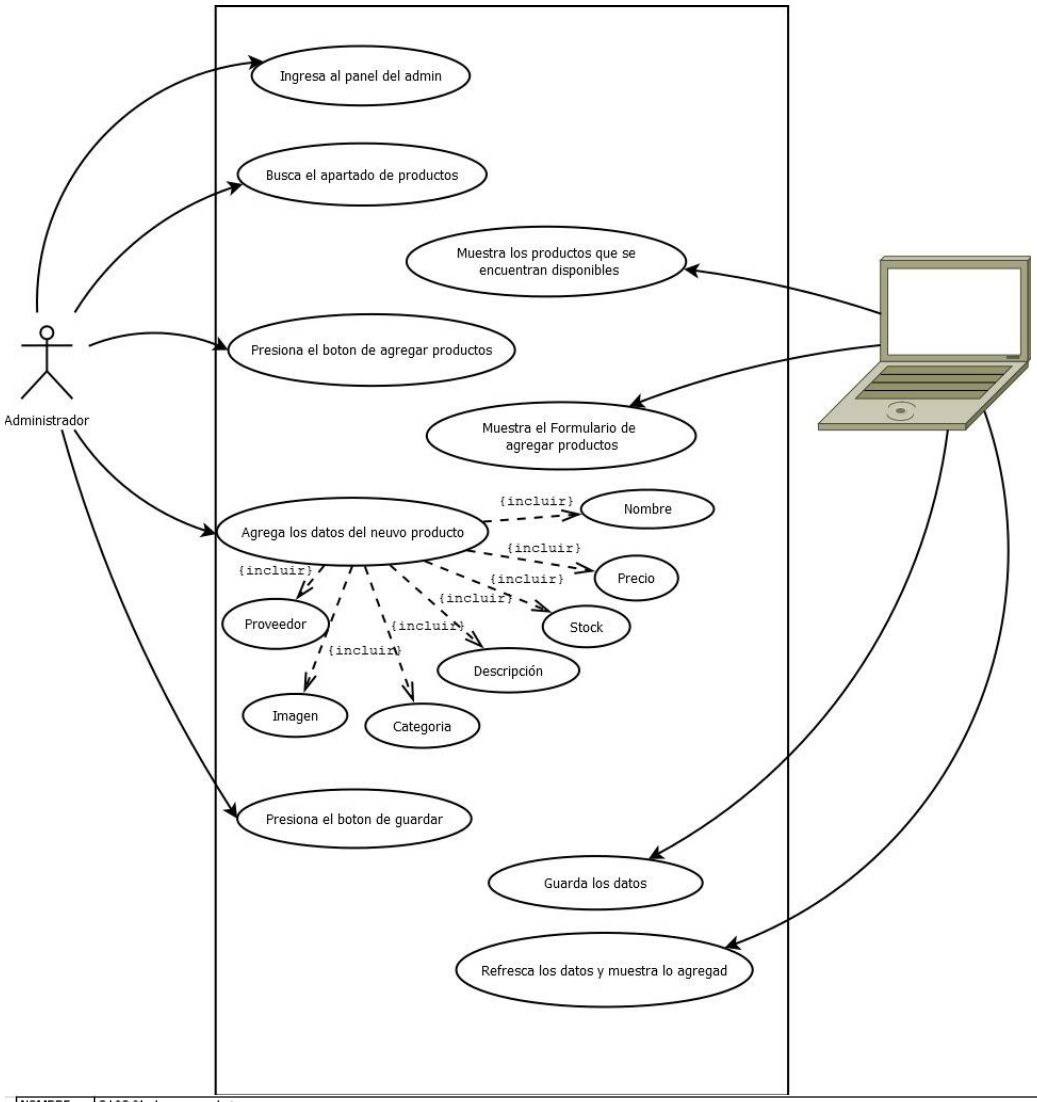
R05

NOMBRE: REGISTRO DE VENTAS



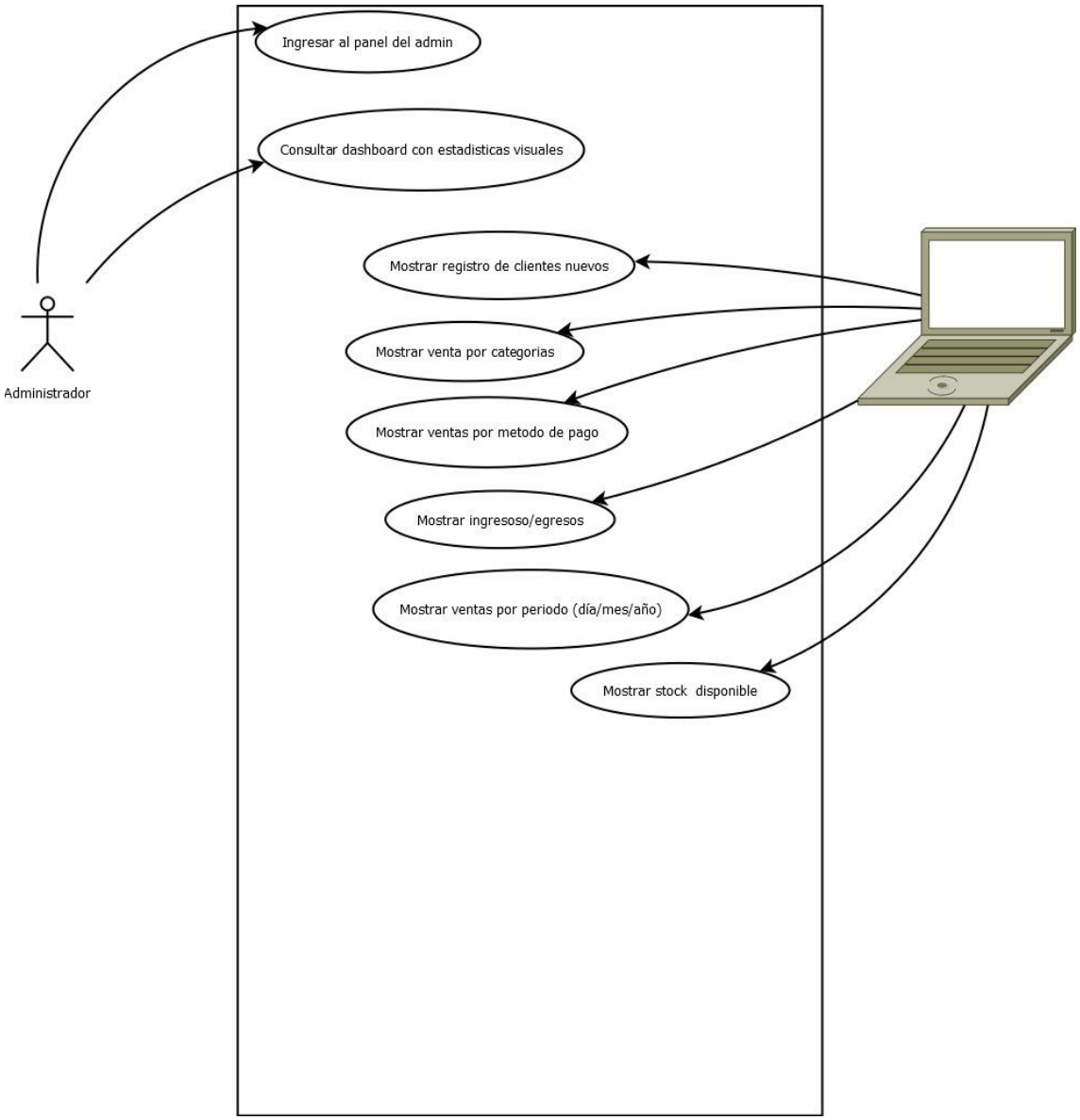
FECHA: 16/09/25		
CASO DE USO 05		
NOMBRE DEL CASO DE USO: REGISTRAR VENTAS		
AUTOR: KEINER LINDARTE.		
ACTOR: USUARIO Y SISTEMA		
PRECONDICION: EL ADMINISTRADOR DEBE TENER UN DISPOSITIVO CON CONEXIÓN A INTERNET DICHO DISPOSITIVOTAMBIEN DEBE TENER ACCESO A LA PÁGINA, ADEMÁS, EL ADMINISTRADOR DEBIÓ HABER INICIADO SECCIÓN YA CON SU RESPECTIVO ROL, LOS PRODUCTOS EXISTEN EN EL INVENTARIO Y TIENEN UN STOCK DISPONIBLE		
FLUJO NORMAL	PASOS	DESCRIPCIONES
	1	El administrador inicia el registro de la venta.
	2	El sistema solicita los datos del cliente.
	3	El administrador selecciona los productos y sus cantidades.
	4	El sistema calcula el monto total.
	5	El administrador confirma la venta y selecciona el método de pago.
	6	El sistema guarda la venta en la tabla buys.
	7	El sistema guarda los detalles de cada producto en la tabla purchasedetails.
	8	El sistema actualiza el stock de los productos vendidos.
	9	El sistema confirma que la venta fue registrada con éxito.
FLUJO ALTERNA	PASOS	DESCRIPCIONES
	2.1	El sistema presenta fallas y no arroja el formulario solicitando los datos del cliente
	4.1	El sistema no genera el monto a pagar
	4.2	El sistema calcula mal el monto a pagar
	6.1	El sistema no logra guardar la venta en su respectiva tabla
	6.2	El sistema falla y no registra el metodo de pago fisico.
	7.1	El sistema no guarda los datos de los productos comprados
	8.1	El sistema no actualiza el stock dejando el mismo sin descontar los productos vendidos.
	9.1	El sistema no arroja el mensaje de confirmación avisnado que la venta fue registrada con éxito.
POSTCONDICION: La venta queda registrada en buys, Los productos vendidos quedan registrados en purchasedetails y El stock de productos se actualiza en products.		

R06



NOMBRE:	CASO 01 - Agregar productos
AUTOR:	Maria Sanchéz
ACTOR :	Admin y Panel de admistracion.

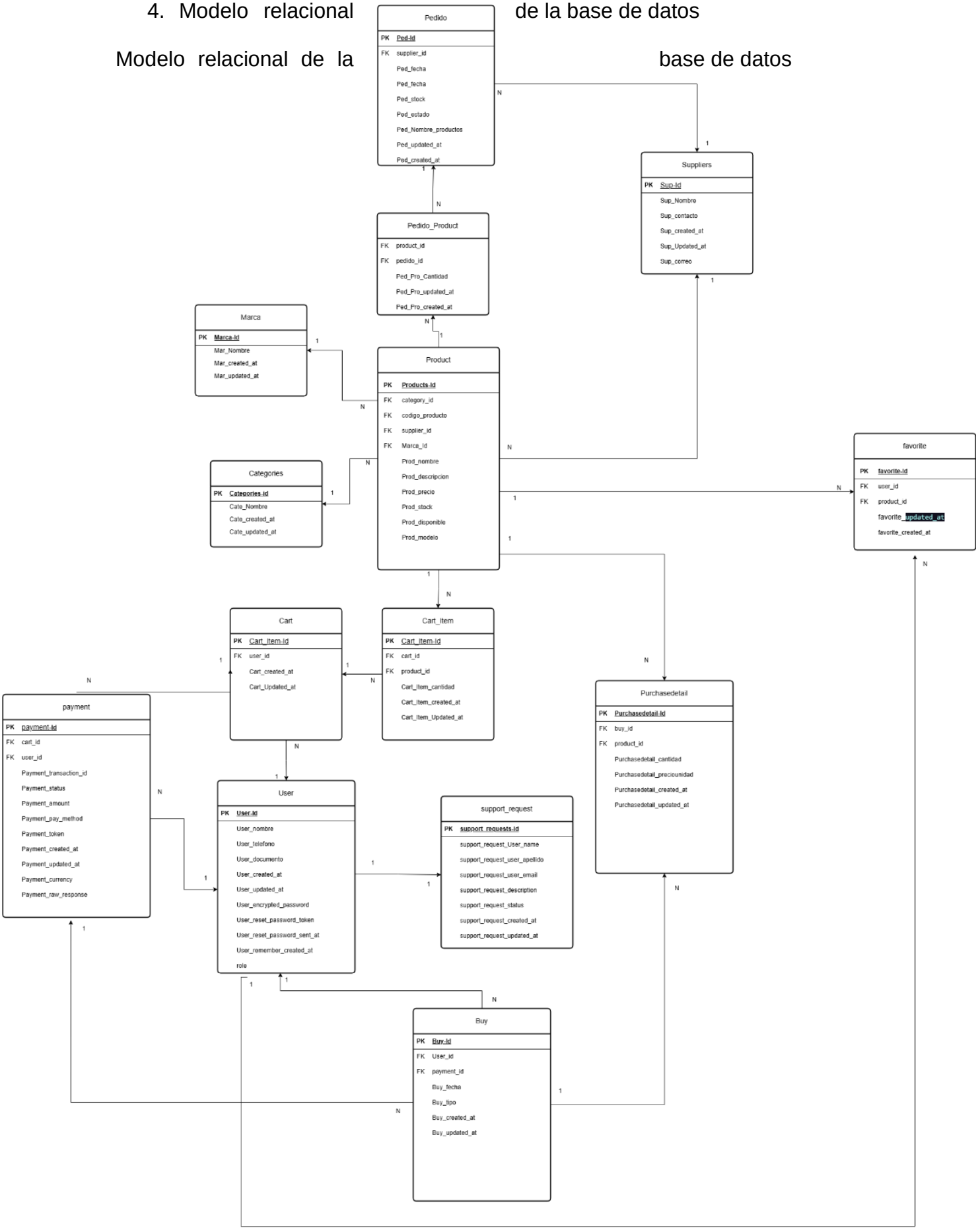
	PRECONDICIÓN
1	El administrador debe estar autenticado en el sistema con permisos de gestión
2	El sistema debe estar disponible y con conexión a la base de datos.
3	Debe existir al menos una categoría registrada y proveedores disponibles para asociar al producto.
4	El administrador debe tener celular o computador con internet.
5	El administrador debe tener la necesidad de agregar un nuevo producto.
	FLUJO NORMAL
1	El administrador ingresa al panel de administración.
2	Navega al apartado de productos.
3	El sistema muestra la lista de productos disponibles en el inventario.
4	El administrador selecciona la opción "Agregar producto".
5	El sistema despliega el formulario de registro de productos.
6	El sistema muestra un formulario con los datos para agregar un nuevo producto (nombre, precio, stock, descripcion, proveedor, categoria, imagen del producto).
7	El administrador llena los datos en los campos y pulsa el boton de "Guardar".
8	El sistema valida los campos ingresados.
9	Si los datos son válidos, el sistema almacena la información en la base de datos.
10	El sistema actualiza la lista de productos y muestra el nuevo producto registrado.
	FLUJO ALTERNATIVO
1.1	El administrador no ingreso correctamente el correo.
1.2	El administrador no ingreso correctamente la contraseña.
6.1	Si el sistema no logra cargar el formulario, se muestra un mensaje de error general y se pide reintentar.
6.2	Si la imagen no cumple con el formato o tamaño permitido, el sistema rechaza la carga y pide una nueva imagen.
7.1	Si el administrador deja un campo obligatorio vacío (nombre, precio, stock, descripcion, proveedor, categoria, imagen del producto el sistema muestra un mensaje de error indicando el campo faltante.
7.2	Si el precio o stock contienen valores no válidos (ejemplo: letras en vez de números, stock negativo), el sistema muestra un error de validación.
7.3	Si el producto ya existe (nombre duplicado), el sistema muestra un mensaje indicando que debe modificar el nombre.
9.1	Si ocurre una falla de conexión con la base de datos, el sistema notifica al administrador que no se pudo guardar y mantiene los datos en pantalla.
9.2	Si el sistema detecta un error interno, muestra un mensaje genérico y cancela el proceso de guardado.
	POSTCONDICIONES
1	El nuevo producto queda registrado correctamente en la base de datos.
2	El sistema actualiza el listado de productos y lo presenta con la información recién ingresada.
3	El producto queda disponible para futuras operaciones (consultas, ventas, reportes, modificaciones).



NOMBRE:	CASO 05 – Visualizar graficas en la dasboard
AUTOR:	Maria Sanchéz
ACTOR:	Admin y Panel de admistracion.
	PRECONDICIÓN
1	El administrador debe estar autenticado en el sistema.
2	Debe existir información en la base de datos (ventas, clientes, productos, pagos, etc.) para generar las estadísticas.
4	El administrador debe tener celular o computador con internet.
	FLUJO NORMAL
1	El administrador ingresa al panel de administración.
2	El sistema consulta los datos en la base de datos
3	El sistema carga la información general de ventas, ingresos, egresos, clientes y stock.
4	El sistema muestra las estadísticas en diferentes gráficos: (Registros de clientes nuevos, ventas por categoría de producto, ingresos y egresos, ventas por método de pago, ventas por día, semana y mes, Nivel de stock de productos)
5	El administrador interpreta la información visual y toma decisiones.
	FLUJO ALTERNATIVO
1.1	El administrador no ingreso correctamente el correo.
1.2	El adminitrador no ingreso correctamente la contraseña.
2.1	Si el sistema no puede conectarse a la base de datos, muestra un mensaje de error: "Error de conexión: no se pudieron obtener los datos. Intente nuevamente más tarde".
3.1	Si no existen ventas registradas, el sistema muestra gráficos vacíos en Ventas por categoría de producto, Ventas por día/semana/mes y Ventas por método de pago, con la leyenda: "Sin información disponible".
3.2	Si no hay registros de clientes, el gráfico de Registros de clientes nuevos aparece en cero y muestra la advertencia: "Aún no se han registrado clientes".
3.3	Si no existen datos financieros, los indicadores de Ingresos y Egresos muestran $0 y se acompaña de la leyenda: "Sin movimientos registrados en este período".
4.1	Si un gráfico específico no puede renderizarse, el sistema muestra "Gráfico no disponible" en su lugar, sin afectar la visualización de los demás.
9.1	Si ocurre una falla de conexión con la base de datos, el sistema notifica al administrador que no se pudo guardar y mantiene los datos en pantalla.
9.2	Si el sistema detecta un error interno, muestra un mensaje genérico y cancela el proceso de guardado.
	POSTCONDICIONES
1	El administrador obtiene una visión general del estado del negocio mediante gráficos y estadísticas.
2	La información presentada sirve como base para decisiones estratégicas (compras, ventas, marketing, control de stock).
3	El sistema actualiza periódicamente los datos para mostrar información confiable.

4. Modelo relacional de la base de datos

Modelo relacional de la base de datos



DICCIONARIO DE DATOS:

cart_items

LLAVE	NOMBRE	TIPO DE DATO	LONGITUD	DESCRIPCIÓN
PK	cart_items_id	integer (autoincrement)	---	Identificador único de la nueva tabla de la relación
FK	cart_id	Integer	---	Identificador único del carrito
FK	product_id	Integer	---	Identificador único del producto
	cart_items_Cantidad	Integer	---	Total de ítems en el carrito de ese usuario en especifico
	cart_items_created_at	datetime	tamaño de almacenamiento fijo	Registro de cuando se hizo el registro.
	cart_items_updated_at	datetime	tamaño de almacenamiento fijo	Registro en fecha y hora de cuando se edita, si las hay.

carts

LLAVE	NOMBRE	TIPO DE DATO	LONGITUD	DESCRIPCIÓN
PK	Id_carts	integer (autoincrement)	---	Identificador único del carrito
FK	user_id	Integer	---	Identificador único del usuario
	Cart_created_at	datetime	tamaño de almacenamiento fijo	Fecha de cuando se hizo el registro.
	Cart_updated_at	datetime	tamaño de almacenamiento fijo	Registro en fecha y hora de cuando se edita, si las hay.

Categories

LLAVE	NOMBRE	TIPO DE DATO	LONGITUD	DESCRIPCIÓN
PK	Id_Categories	Integer	---	Número que identifica la categoría
	Categories_nombre	String	50	Nombre de la categoría
	Categories_created_at	datetime	tamaño de almacenamiento fijo	Fecha de cuando se creó el registro.
	Categories_updated_at	datetime	tamaño de almacenamiento fijo	Registro en fecha y hora de cuando se edita, si las hay.

Users

LLAVE	NOMBRE	TIPO DE DATO	LONGITUD	DESCRIPCIÓN
PK	Id_Users	integer (autoincrement)	---	Número que identifica al usuario
	User_Nombre	String	60	Nombre de quien realizó la compra
	User_Teléfono	String	60	Número de contacto del cliente
	User_created_at	datetime	tamaño de almacenamiento fijo	Fecha de cuando se creó el registro.
	User_updated_at	datetime	tamaño de almacenamiento fijo	Registro en fecha y hora de cuando se edita, si las hay.
	User_ email	string	50	Dirección de correo electrónico del usuario registrado
	encrypted_password	string	255	Contraseña encriptada
	reset_password_token	string	100	Token de recuperación

	User_Role	string	20	Rol (User, admin, etc)

Favorites

LLAVE	NOMBRE	TIPO DE DATO	LONGITUD	DESCRIPCIÓN
PK	Favorite_Id	Integer	---	Identificador único del favorito
FK	product_id	Integer	---	Identificador único del producto
FK	user_id	Integer	---	Identificador único del usuario
	created_at	Datetime	tamaño de almacenamiento fijo	Fecha de creación del registro
	updated_at	Datetime	tamaño de almacenamiento fijo	Fecha de última actualización

Marcas

LLAVE	NOMBRE	TIPO DE DATO	LONGITUD	DESCRIPCIÓN
PK	Marca_Id	integer (autoincrement)	---	Identificador único de la marca
	Marca_nombre	String	60	Nombre de la marca del producto
	Marca_created_at	Datetime	tamaño de almacenamiento fijo	Fecha de creación del registro
	Marca_updated_at	Datetime	tamaño de almacenamiento fijo	Fecha de última actualización

Payments

LLAVE	NOMBRE	TIPO DE DATO	LONGITUD	DESCRIPCIÓN
PK	Pay_Id	integer (autoincrement)	----	Identificador único del pago
FK	cart_id	Integer	----	ID del carrito asociado (relación con tabla carts)
FK	user_id	Integer	----	
	transaction_id	String	50	ID único generado por Wompi
	status	integer (enum)	----	Estado del pago (pending, approved, failed, paid)
	Amount	Decima	precision: 12, scale: 2	Monto total de la transacción. Hasta 999,999,999,999 .99 COP
	pay_method	String	30	Tipo de pago/ meotdo de pago (CARD, PSE, NEQUI, etc)
	Pay_token	String	100	Token temporal del pago.
	Pay_Currency	String	3	Código de onesa ISO 4217
	Wompi_Id	String	50	ID único del pago dentro de wompi.
	Pay_raw_response	Json	estructura variable según respuesta de pasarela	Respuesta completa en formato JSON de la pasarela de pago
	Pay_account_info	String	150	Información asociada a la cuenta o método de pago (últimos

				dígitos, nombre, etc.)
	Pay_created_at	Datetime	tamaño de almacenamiento fijo	Fecha de creación del registro
	Pay_updated_at	Datetime	tamaño de almacenamiento fijo	Fecha de última actualización

products

LLAVE	NOMBRE	TIPO DE DATO	LONGITUD	DESCRIPCIÓN
PK	Products_Id	Integer	----	codigo_producto
	category_id	Integer	----	
	supplier_id	Integer	----	
	marca_id	Integer	----	
	Pro_nombre	String	100	Nombre del producto
	Pro_descripcion	String	200	Descripción breve
	Pro_ precio	decimal	12,2	Precio del producto
	Pro_stock	Integer	----	Cantidad en la tienda de dicho producto
	Pro_ modelo	String	50	Modelo o referencia
	Pro_ disponible	Boolean	----	Indica si está disponible
	Pro_created_at	Datetime	tamaño de almacenamiento fijo	Fecha de creación del registro
	Pro_updated_at	Datetime	tamaño de almacenamiento fijo	Fecha de última actualización

user_supports

LLAVE	NOMBRE	TIPO DE DATO	LONGITUD	DESCRIPCIÓN
PK	User_Id	integer (autoincrement)	---	Identificador único del usuario
	User_ User_name	String	100	Nombre del usuario que envía la solicitud
	user_apellido	string	100	Apellido del usuario que envía la solicitud
	user_email	String	100	Correo del usuario
	User_description	Text	----	Descripción detallada del problema
	User_status	String	30	Estado (pendiente, resuelto, etc.)
	Pay_created_at	Datetime	tamaño de almacenamiento fijo	Fecha de creación del registro
	Pay_updated_at	Datetime	tamaño de almacenamiento fijo	Fecha de última actualización

Suppliers

LLAVE	NOMBRE	TIPO DE DATO	LONGITUD	DESCRIPCIÓN
PK	Sup_Id	Integer	50	
	Sup_ nombre	string	100	Nombre del proveedor
	Sup_contacto	string	100	Nombre de contacto
	Sup_ codigo_proveedor	string	30	Código interno del proveedor
	Sup_ correo	string	100	Correo electrónico del proveedor

5. Descripción de la plataforma

El sistema de información para la ferretería fue desarrollado como una aplicación web basada en arquitectura MVC (Modelo-Vista-Controlador), utilizando tecnologías modernas que garantizan escalabilidad, seguridad y facilidad de mantenimiento.

En cuanto al Lenguaje de programación utilizamos Ruby (versión 3.x): Lenguaje de programación orientado a objetos y muy utilizado para aplicaciones que se emplea en combinación con el framework Rails para la construcción del backend y la lógica de negocio.

En cuanto al framework principal como ya se mencionó se utilizó Ruby on Rails (versión 7.x): Un Framework de desarrollo web de código abierto que sigue el patrón MVC. Proporciona estructura, herramientas integradas y un entorno robusto para construir aplicaciones web de manera rápida y ordenada.

Sistema de gestión de bases de datos (SGBD) PostgreSQL (en producción) / SQLite3 (en desarrollo local): Base de datos relacional utilizada para almacenar información de productos, compras, ventas, proveedores, usuarios, etc.

Se accede mediante Active Record, el ORM nativo de Rails.

Frontend (interfaz de usuario): HTML5, CSS3, ERB (Embedded Ruby), Bootstrap 5, Framework CSS para diseño responsivo.

Permite una interfaz moderna, adaptable a dispositivos móviles.

Se utilizaron componentes dinámicos como modales, tarjetas, botones flotantes, formularios interactivos, etc.

Devise fue una gema utilizada para el Manejo de autenticación de usuarios (login, logout, sesión segura).

gem "rails-i18n" internacionalización

Los Bootstrap Modals son ventanas emergentes personalizadas para registrar proveedores, productos y otras acciones sin recargar la página.

Entorno de ejecución y servidor: Rails Server (Puma): Servidor web predeterminado de Rails.

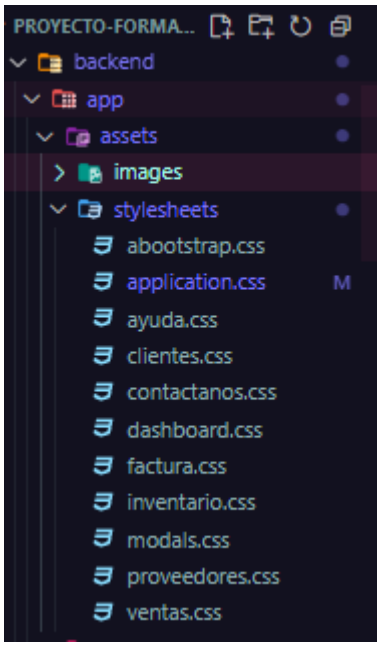
Sistema operativo compatible: Windows 10/11 para desarrollo local

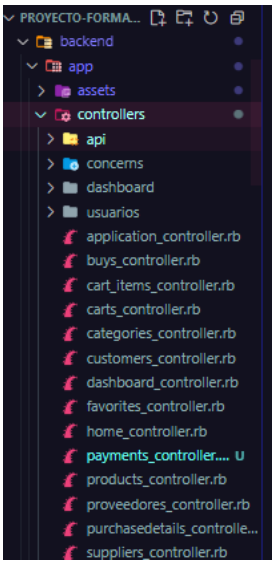
Control de versiones

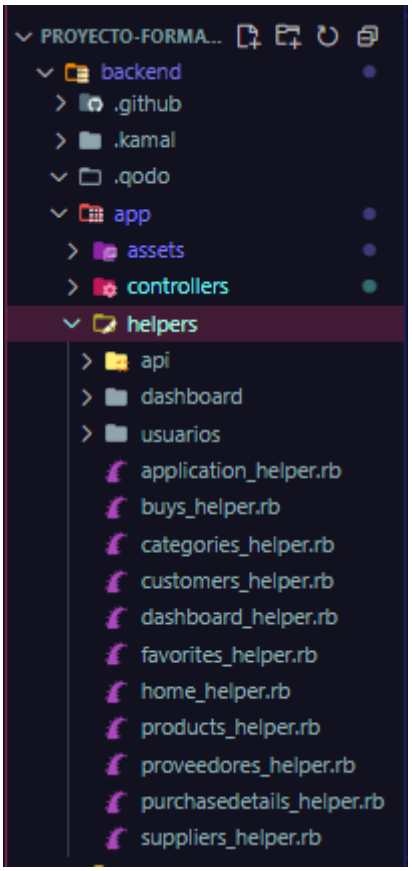
GitHub: Almacenamiento del código fuente y colaboración en el desarrollo.

6. Documentación del código fuente

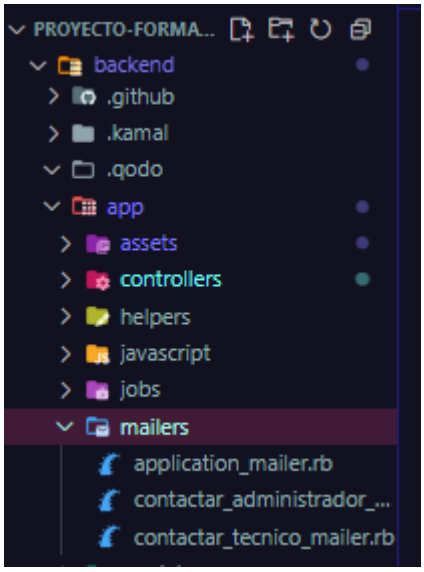
ESTILOS CSS CONTROLADORES HELPERS

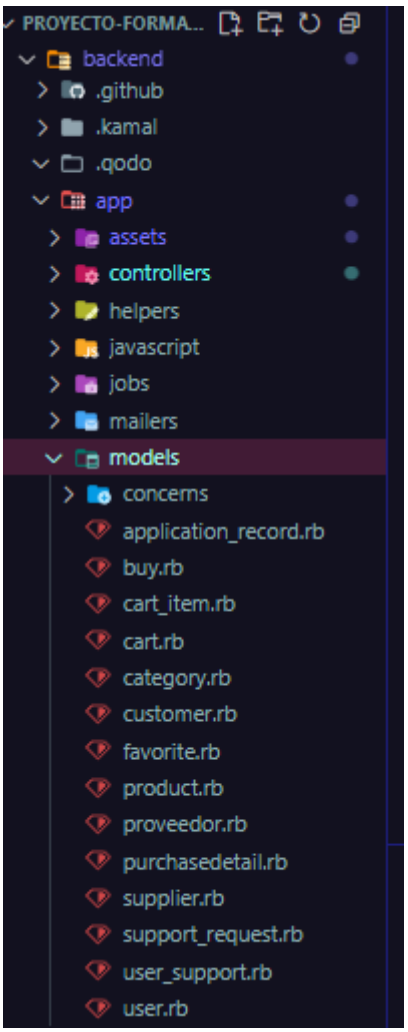


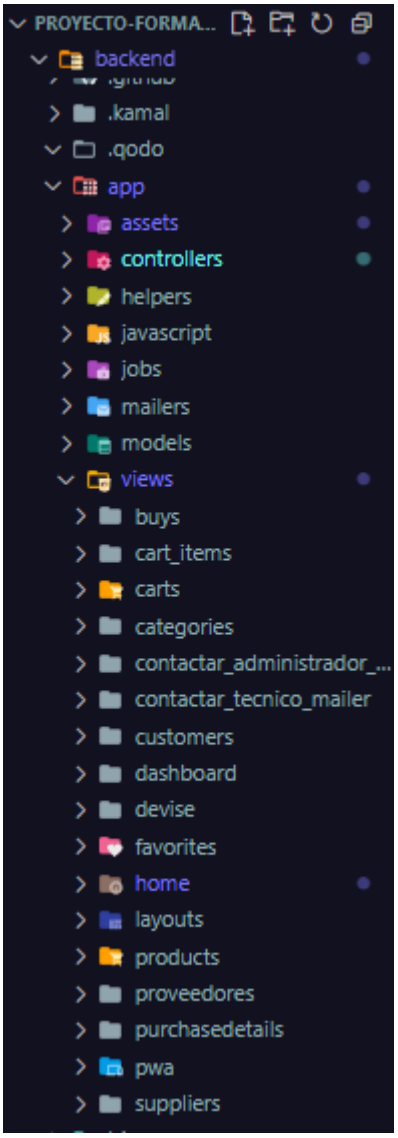




MAILERS **MODELOS** **VIWS O VISTAS**





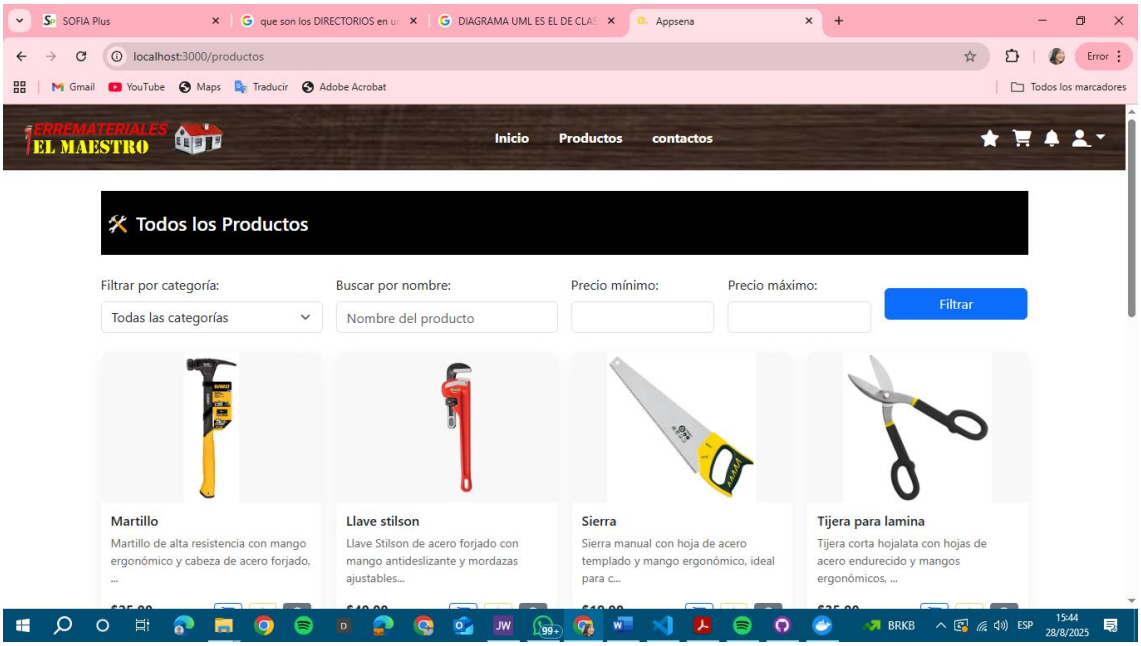


2. Estructura de directorios:

DIRECTORIO	DESCRIPCIÓN
app/controllers/	Guarda los controladores, que manejan la lógica de la aplicación y el flujo de datos entre las vistas y los modelos.
app/models/	Alberga los modelos, que representan la estructura de los datos de la base de datos y la lógica de negocio.
app/ assets /	Contiene los archivos de vista, que son responsables de presentar la interfaz de usuario.
App/assets/ stylesheets	Contiene los archivos de estilos css que le dan color y vida al código.
app/ assets / images	Contiene las imágenes usadas en la aplicación.

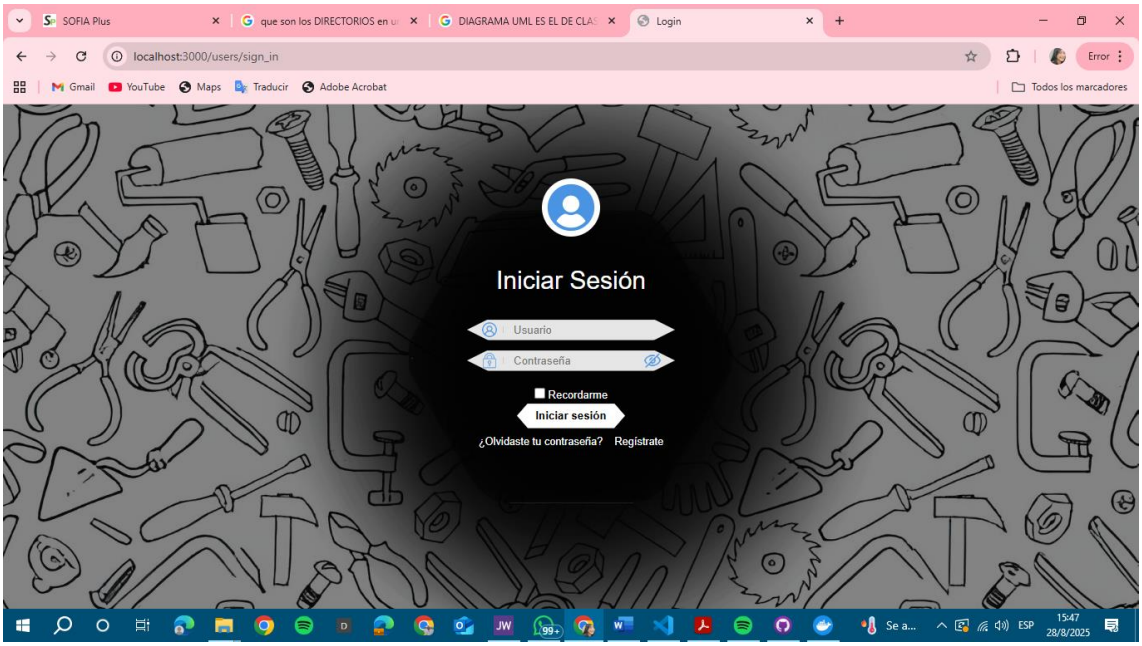
app/helpers/	se utiliza para funciones auxiliares reutilizables en las vistas.
app/mailers/	Contiene la lógica y las plantillas para enviar correos electrónicos desde la aplicación.

VISTA DE PRODUCTOS DE USUARIOS:




```
backend > app > views > home > index.html.erb
<!-- PRODUCTOS -->
<section id="productos" class="productos-destacados container mt-4 mb-5">
  <h2 class="titulo">🔥 Productos destacados</h2>
  <div class="product-grid">
    <% @productos.each do |producto| %>
      <div class="product-card">
        <div class="product-thumb">
          <%= image_tag(producto.imagen.presence || "martillo-dewalt.png",
                        alt: producto.nombre,
                        class: "img-fluid") %>
        </div>
        <div class="product-body">
          <h3 class="product-name"><%= producto.nombre %></h3>
          <p class="product-desc"><%= truncate(producto.descripcion, length: 80) %></p>
          <div class="product-meta">
            <span class="price">
              <%= number_to_currency(producto.precio, unit: "$", separator: ".", delimiter: ",") %>
            </span>
            <div>
              <div class="d-flex flex-wrap gap-2 mt-2">
                <!-- Botón comprar -->
                <%= button_to cart_items_path(product_id: producto.id),
                method: :post,
                remote: true,
                class: "btn btn-sm btn-outline-primary d-flex align-items-center justify-content-center" do %>
                  <i class="fa-solid fa-cart-plus"></i>
                <% end %>
```

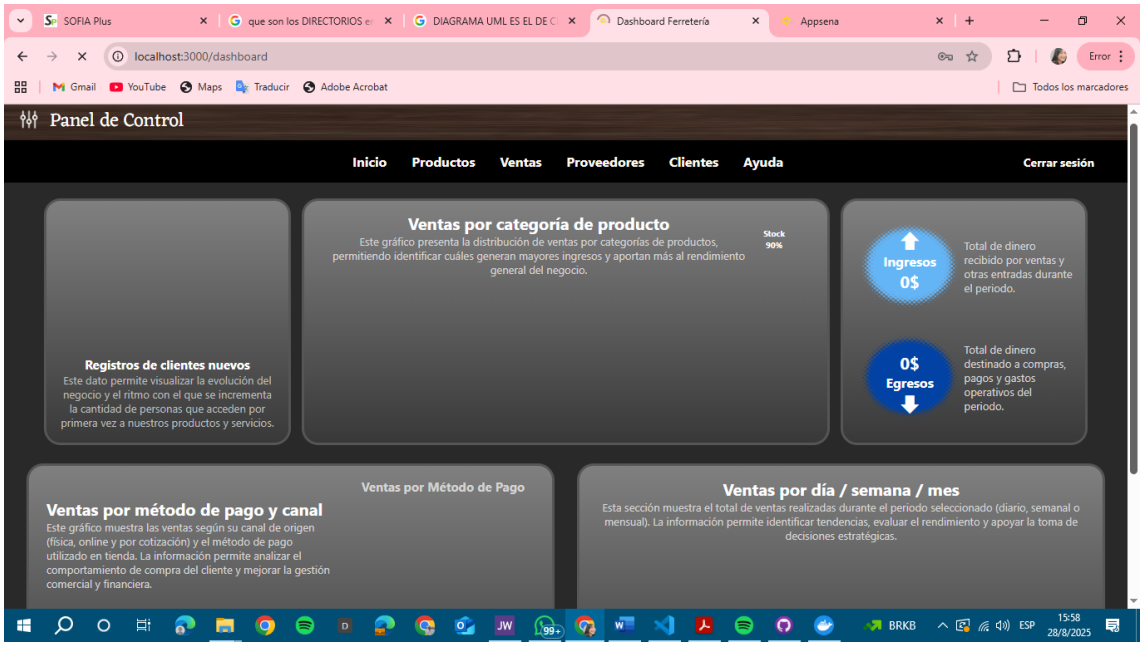
LOGIN




```
    <div class="input-wrapper">
      <%= image_tag "candado.png", class: "icon-img", alt: "Contraseña" %>
      <div class="divider"></div>
      <%= f.password_field :password, placeholder: "Contraseña", autocomplete: "current-password", id: "password" %>
      <%= image_tag "ojo.png", id: "eye-icon", class: "toggle-password eye-img", onclick: "togglePassword()", alt: "Ver contraseña" %>
    </div>

    <% if devise_mapping.rememberable? %>
      <div class="field mb-1 form-check text-start">
        <%= f.check_box :remember_me, class: "form-check-input" %>
        <%= f.label :remember_me, "Recordarme", class: "form-check-label" %>
      </div>
    <% end %>

    <div class="hex-button-wrapper mb-3">
      <%= f.submit "Iniciar sesión", class: "hex-button" %>
    </div>


  <% end %>

  <!-- Links al fondo -->
  <div class="bottom-links">
    <%= render "devise/shared/links" %>
  </div>



  </div>
</div>
</div>
```

```
<div class="login d-flex justify-content-center align-items-center vh-100">

  <div class="hex-wrapper position-relative">

    <!-- Imagen del hexágono como fondo -->
    <%= image_tag "hexagono.png", alt: "Hexágono", class: "hex-bg" %>

    <!-- Icono del usuario superpuesto -->
    <div class="user-icon position-absolute top-0 start-50 translate-middle">
      <%= image_tag "user.png", alt: "Usuario", class: "rounded-circle user-img" %>
    </div>

    <!-- Contenido del login sobre el hexágono -->
    <div class="hex-content position-absolute top-50 start-50 translate-middle text-center">
      <% resource = User.new %>
      <% resource_name = :user %>
      <% devise_mapping = Devise.mappings[:user] %>

      <!-- Mensajes de error -->
      <%= render "devise/shared/error_messages", resource: resource %>

      <%= form_for(resource, as: resource_name, url: session_path(resource_name)) do |f| %>
        <div class="text-sg">
          <%= "Iniciar Sesión" %>
        </div>

        <div class="input-wrapper">
          <%= image_tag "avatar.png", class: "icon-img", alt: "Usuario" %>
          <div class="divider"></div>
          <%= f.email_field :email, placeholder: "Usuario", autocomplete: "email" %>
        </div>
```

DASHBOARD




```erb
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Dashboard Ferretería</title>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/chartjs-plugin-datalabels"></script>
  <script src="https://cdn.jsdelivr.net/npm/chartjs-chart-radial-gauge@^0.3.0/dist/chartjs-chart-radial-gauge.min.js"></script>
  <%= favicon_link_tag 'favicon.png' %>
  <%= stylesheet_link_tag "dashboard", media: "all" %>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.0/css/all.min.css">
  <link href="https://fonts.googleapis.com/css2?family=Eczar:wght@400;500;600;700;800&display=swap" rel="stylesheet">
  <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@700&display=swap" rel="stylesheet">

  <%= javascript_include_tag "application", "data-turbo-track": "reload", defer: true %>
  <%= javascript_importmap_tags %>
</head>
<body>

  <header>
    <div class="container-header">
      <button class="hamburger" id="hamburger">&#9776;</button>
      <div id="control">
        <div class="panel"><%= image_tag "panel-control.png", class: "c-panel" %></div>
        <p class="pricipal">Panel de Control</p>
      </div>
      <% if user_signed_in? %>
        <%= button_to destroy_user_session_path, method: :delete, form: { data: { turbo: true } }, id: "cerrar", class: "logout2" do %>
          <span><i class="fas fa-sign-out-alt"></i></span>
        <% end %>
      <% end %>
    </div>
  </header>
```

```
<nav class="nav-bg">
  <div class="container-nav">
    <ul class="menu" id="menu">
      <li><a href="dashboard"class="link-menu">Inicio</a></li>
      <li><a href="dashboard/productos"class="link-menu">Productos</a></li>
      <li><a href="dashboard/ventas"class="link-menu">Ventas</a></li>
      <li><a href="dashboard/suppliers"class="link-menu">Proveedores</a></li>
      <li><a href="dashboard/clientes"class="link-menu">Clientes</a></li>
      <li><a href="dashboard/help"class="link-menu">Ayuda</a></li>
    </ul>
    <% if user_signed_in? %>
      <%= button_to "Cerrar sesión", destroy_user_session_path, method: :delete, form: { data: { turbo: true } }, id: "cerrar", class: "logout"%
    <% end %>
  </div>
</nav>
<section class="fondo">
  <section class="section-1">
    <div class="grafica-1">
      <section class="graf-1">
        <canvas id="graficoClientesMes"></canvas>
      </section>
      <section class="texto-1">
        <div class="titulo-1">Registros de clientes nuevos</div>
        <div class="texto-1-1">Este dato permite visualizar la evolución del negocio y el ritmo con el que se incrementa la cantidad
          de personas que acceden por primera vez a nuestros productos y servicios.</div>
      </section>
    </div>
    <div class="grafica-2">
      <section class="texto-2">
        <section class="texto-2-1"><div class="titulo-2">Ventas por categoría de producto</div>Este gráfico presenta la distribución de
          ventas por categorías de productos, permitiendo identificar cuáles generan mayores ingresos y aportan más al rendimiento general del n
        </section>
        <div class="img-2">
          <canvas id="gaugeSegmentado"></canvas>
```

```
  <section class="fondo">
    <section class="section-1">
      <div class="grafica-2">
        <section class="texto-2">
          </div>
        </section>
        <section class="graf-2">
          <canvas id="graficoVentas" ></canvas>
        </section>
      </div>
      <div class="grafica-3">
        <section>
          <div class="ingresos">
            <p class="flecha" id="F1">⬆</p>
            <p class="etiqueta">Ingresos</p>
            <p class="monto" id="ingresos">0$</p>
          </div>
          <!--<div class="contenedor-moneda">
            <div class="texto-centro">
              <div class="flecha">↑</div>
              <div class="etiqueta">Ingresos</div>
              <div class="monto">$3.250.000</div>
          </div>-->
          <div class="texto-3">Total de dinero recibido por ventas y otras entradas durante el periodo.</div>
        </section>
        <section>
          <div class="egresos">
            <p class="monto" id="egresos">0$</p>
            <p class="etiqueta">Egresos</p>
            <p class="flecha" id="F2">⬇</p>
          </div>
          <div class="texto-3">Total de dinero destinado a compras, pagos y gastos operativos del periodo.</div>
        </section>
      </div>
```

```
<!-- grfaica de la seccion 1 -->
<script>
  document.addEventListener('DOMContentLoaded', () => {
    fetch('/api/v1/clientes_por_mes')
      .then(res => res.json())
      .then(data => {
        const meses = [
          'Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio',
          'Julio', 'Agosto', 'Septiembre', 'Octubre', 'Noviembre', 'Diciembre'
        ];

        const valores = Object.values(data);
        const etiquetas = Object.keys(data).map(num => meses[parseInt(num) - 1]);

        const colores = [
          '#1e88e5', // Enero
          '#103f91', // Febrero
          '#1565c0', // Marzo
          '#1976d2', // Abril
          '#2196f3', // Mayo
          '#3399ff', // Junio
          '#42a5f5', // Julio
          '#1e88e5', // Agosto
          '#64b5f6', // Septiembre
          '#90caf9', // Octubre
          '#bbdefb', // Noviembre
          '#e3f2fd'  // Diciembre
        ];

        const ctx = document.getElementById('graficoClientesMes').getContext('2d');
```

```
<script>
  document.addEventListener('DOMContentLoaded', () => {
    fetch('/api/v1/clientes_por_mes')
      .then(data => {

        const ctx = document.getElementById('graficoClientesMes').getContext('2d');

        new Chart(ctx, {
          type: 'doughnut',
          data: {
            labels: etiquetas,
            datasets: [{
              data: valores,
              backgroundColor: colores,
              borderColor: '#bbbbbbff',
              borderWidth: 2,
              cutout: '40%'
            }]
          },
          options: {
            plugins: {
              legend: { display: false },
              datalabels: {
                color: 'white',
                font: {
                  weight: 'bold',
                  size: 12
                },
                formatter: (value, context) =>
                  context.chart.data.labels[context.dataIndex] + ': ' + value
              }
            }
          },
          plugins: [ChartDataLabels]
        });
```

7. Descripción de los acuerdos de niveles de servicios o ANS

1. Mesa de ayuda y soporte técnico

El sistema contempla una sección de ayuda dentro de la dashboard donde el usuario puede acceder a documentación básica del sistema (guías, manuales o preguntas frecuentes).

En caso de requerirse soporte adicional, se contemplan los siguientes canales:

Canal Función

Documentación interna:

- Manual técnico y guía de uso
- Soporte por correo.
- Atención personalizada en caso de error crítico.
- Soporte técnico local Soporte del desarrollador del sistema o equipo TI.

TIEMPO DE RESPUESTAS (SLAs)

Errores críticos del sistema:

0 – 6 horas Alta

Fallos menores: (estéticos, textos, etc.) 24 – 48 horas Media

Nuevas funcionalidades o mejoras: 3 – 7 días hábiles Baja

Consultas o preguntas técnicas:

12 – 24 horas Media

El nivel de soporte es el Nivel 1 Usuario administrador (soporte básico) Consulta documentación, verifica errores simples.

Nivel 2 Soporte técnico / desarrollador Soluciona errores de lógica, configuración o base de datos.

Nivel 3: Mantenimiento avanzado (si aplica) Aplicación de parches, migraciones, escalamiento técnico.

Protocolo de escalamiento:

El administrador detecta el problema y revisa documentación o intenta solución básica.

Si no se resuelve, se reporta el incidente al soporte técnico mediante los canales acordados.

Si el soporte de Nivel 2 no puede resolverlo, se escala a Nivel 3 para intervención avanzada.

4. Mantenimiento y actualizaciones

Las actualizaciones del sistema pueden incluir mejoras visuales, nuevos reportes o ajustes en la lógica de negocio.

Se recomienda agendar mantenimientos fuera del horario operativo para evitar interrupciones.

8. **Administración de usuarios**

se implementa una autenticación simple centrada en el rol de administrador, quien es el único usuario autorizado a ingresar al panel de control (dashboard) de la ferretería.

Modificación de usuarios:

- Se puede implementar en la consola Rails.
- Permite actualizar correo electrónico, contraseña u otros atributos.

Retiro o desactivación:

- Actualmente no se contempla eliminación desde la interfaz, sin embargo, se puede eliminar o desactivar una cuenta directamente en la base de datos.

b. Creación de roles y perfiles de usuario

El sistema cuenta actualmente con dos roles activo:

Administrador: Tiene acceso completo al dashboard, puede gestionar productos, ventas, compras, clientes y proveedores.

Visitantes:

No requieren autenticación, pueden: Navegar el catálogo de productos, enviar mensajes mediante el formulario de contacto y también puede solo si lo desea registrarse para tener una cuenta propia en donde podrá administrar su información, hacer pedidos y demás.

GLOSARIO:

Framework: Un framework, o marco de trabajo, es un conjunto de herramientas, bibliotecas, estructuras y convenciones estandarizadas que proporciona una base para desarrollar software o aplicaciones, simplificando el proceso y evitando empezar desde cero.

Active Record: patrón de diseño y un enfoque de programación que permite representar una tabla de base de datos como una clase de objeto, donde cada fila de la tabla se corresponde con una instancia de esa clase.

Frontend: Parte de un sitio web o aplicación con la que los usuarios interactúan directamente, es decir, todo lo que el usuario ve y experimenta en su navegador o dispositivo

Backend: es la parte "detrás de escena" de una aplicación o sitio web, responsable de la lógica, el funcionamiento y los datos que no son visibles para el usuario final. Se encarga de las tareas cruciales como la gestión de bases de datos, el procesamiento de información, la seguridad, la autenticación de usuarios y la comunicación con el servidor

Html: el estándar para crear y estructurar el contenido de las páginas web. A través de etiquetas y elementos, HTML define la estructura del contenido, como párrafos, imágenes, enlaces y tablas, indicando a los navegadores cómo organizar y presentar la información al usuario.

API: Una API (Interfaz de Programación de Aplicaciones) es un conjunto de reglas y protocolos que permite a diferentes aplicaciones de software comunicarse entre sí, intercambiar datos y utilizar funcionalidades. Actúa como un intermediario o puente que define cómo se realizan las solicitudes y las respuestas, simplificando el desarrollo al permitir a los programadores usar funciones existentes en lugar de crearlas desde cero.

CSS (Hojas de Estilo en Cascada): Es un lenguaje utilizado para definir la apariencia y el diseño visual de los elementos en un documento, principalmente en páginas web. Permite controlar aspectos como el color, la tipografía, el tamaño, el espaciado y la disposición de los elementos, separando la estructura (HTML) de la presentación, lo que facilita la creación de sitios web atractivos y con un código más limpio y organizado.

BASE DE DATOS: Colección organizada y estructurada de información, o datos, que se almacena y gestiona electrónicamente en un sistema informático, controlada generalmente por un Sistema de Gestión de Bases de Datos (SGBD). Las bases de datos en programación permiten a los desarrolladores almacenar, consultar, modificar y administrar de forma eficiente grandes volúmenes de datos para crear aplicaciones, analizar información o tomar decisiones informadas.

BASE DE DATOS RELACIONAL: Es un sistema que organiza y almacena datos en una o más tablas (llamadas también "relaciones") compuestas por filas y columnas. Permite relacionar datos entre tablas, lo que facilita la búsqueda y el análisis de la información, utilizando un lenguaje estandarizado llamado SQL para interactuar con los datos.

MODELO DE DATOS: Representación visual de un sistema de información, que define las estructuras de datos, los atributos que contienen y las relaciones entre ellas, facilitando la organización, comprensión y gestión de la información para el desarrollo de sistemas y bases de datos.

UML (Lenguaje Unificado de Modelado): Es un lenguaje gráfico estandarizado y de propósito general que se usa para especificar, visualizar, construir y documentar sistemas de software, no para programar directamente. A través de sus diferentes diagramas (estructurales y de comportamiento), permite modelar la arquitectura, las relaciones y el comportamiento de un sistema, facilitando la comprensión y comunicación de diseños complejos entre los desarrolladores y otros miembros del equipo.

DIAGRAMA DE CLASE: Representación visual, utilizando el lenguaje UML (Lenguaje Unificado de Modelado), que muestra la estructura estática de un sistema de software, detallando sus clases, los atributos (datos) y operaciones (métodos) que definen a cada clase, así como las relaciones (como herencia, asociación o composición) que existen entre ellas. Es una herramienta fundamental en la ingeniería de software para entender, diseñar y comunicar la arquitectura de un sistema de forma clara y concisa.

DIAGRAMA DE CASO DE USO: representación gráfica que muestra las funcionalidades de un sistema desde la perspectiva del usuario final, identificando los actores (usuarios o sistemas externos) y los casos de uso (las tareas que el sistema puede realizar)

herramienta de modelado visual que muestra las interacciones entre los usuarios (actores) y las funcionalidades (casos de uso) de un sistema

FICHA DE CASO DE USO: documento detallado que describe cómo un usuario interactúa con un sistema para alcanzar un objetivo específico.

Documento o plantilla textual que describe una funcionalidad específica de un sistema, detalla cómo un actor (usuario o sistema externo) interactúa con el sistema para lograr un objetivo definido.

MVC (Modelo-Vista-Controlador): Es un patrón de arquitectura que divide una aplicación en tres componentes principales para una mejor organización del código y un desarrollo más mantenible. El Modelo gestiona los datos y la lógica de negocio, la Vista se encarga de la interfaz de usuario y de presentar la información, y el Controlador actúa como intermediario, enrutando las solicitudes del usuario al Modelo y luego a la Vista.

PostgreSQL: Sistema avanzado de gestión de bases de datos relacionales (RDBMS) de código abierto que se utiliza para almacenar, organizar y gestionar grandes cantidades de datos de manera confiable y eficiente. Soporta tanto datos estructurados como no estructurados (como JSON), ofrece características de alto rendimiento, escalabilidad, y es compatible con los principales sistemas operativos y lenguajes de programación.