**Para Empezar:**

Para desarrollar un sistema de ventas sencillo utilizando PHP ,javaScript Bootstraap y DataTables,es esencial seguir un proceso estructurado que incluya la recopilación de requerimientos, análisis de software, diseño y arquitectura, documentación y pruebas. Continuación, te proporcionamos una guía detallada para cada una de estas etapas:

**Preguntas Frecuentes Para Reunir Requerimientos**

**Preguntas Generales:**

* ¿Cuál es el propósito principal del sistema de ventas de automóviles?
  + El propósito principal de un sistema de ventas de automóviles es facilitar y optimizar el proceso de venta y gestión de automóviles en una concesionaria o empresa de venta de autos y camionetas. Este sistema permite gestionar de manera eficiente las diversas actividades relacionadas con la venta de vehículos, incluyendo:

1. **Inventario:** Mantener un registro actualizado de los autos disponibles, sus características, estado.
2. **Cliente:**  Gestionar la información de los clientes, sus preferencias.
3. **Ventas:** Atomizar y simplificar el proceso de venta.
4. **Finanzas:** Manejar pagos, financiamientos, descuentos y promociones de manera eficiente.
5. **Reportes:** Generar reportes y análisis para tomar decisiones informadas y mejorar la estrategia de ventas.

En resumen, el sistema busca incrementar la eficiencia, reducir errores, mejorar la experiencia del cliente y, en última instancia, aumentar las ventas y la rentabilidad de la empresa.

* ¿Quiénes serán los usuarios principales del sistema?
  1. **Vendedores:** Utilizan el sistema para gestionar las cotizaciones, seguimientos, cierres de ventas.
  2. **Gerentes de Ventas:** Supervisan las actividades de los vendedores, analizan el rendimiento de ventas.
  3. **Personal de Inventario**: Mantienen actualizado el inventario de vehículos, gestionan la entrada y salida de autos y controlan el stock disponible.
  4. **Personal de Finanzas:** Manejan los aspectos financieros de las ventas, incluyendo pagos, descuentos y promociones.
  5. **Administradores del Sistema:** Gestionan la configuración, seguridad y mantenimiento del sistema, garantizando su correcto funcionamiento.

Estos usuarios interactúan con el sistema para asegurar una operación eficiente y coordinada en todas las etapas del proceso de ventas de automóviles.

* ¿Qué tipo de productos o servicios se venderán a través del sistema?
  1. **Vehículos menores:** Automóviles entre ello autos y camionetas nuevos de diversas marcas, modelos y características.
     + Marcas:
       - Jac
       - Changan
       - Chery
       - Toyota
       - Hyundai
       - Kia
       - Chevrolet
       - Nissan
       - Suzuki
       - Audi
       - BMW
       - BYD
       - Dodge
       - Ford
       - Mercedes Benz
       - Mitsubishi
       - Peugeot
       - Porsche
     + Modelos:
       - Sedan
       - Station wagon
       - Suv
       - Todo Terreno
       - Camionetas
       - Pick up
       - Autos Eléctricos

Estos productos y servicios complementan la experiencia de compra de automóviles, ofreciendo opciones adicionales y facilitando las necesidades y preferencia de los clientes.

**Funcionalidades:**

* ¿Qué características específicas debe tener el sistema de ventas de automóviles?
  1. **Gestión de Inventario: Mantener** un registró detallado y actualizado de todos los vehículos disponibles, incluyendo caracteristicas,estado y precios.
  2. **Procesamiento de Ventas:** Facilitar el proceso de venta desde la cotización, asegurando la precisión y eficiencia en la gestión de venta y documentación.
  3. **Generación de Finanzas:** Administrar aspectos financieros como descuentos, promociones y pagos pendientes.
  4. **Seguridad y Acceso Controlado:** Implementar medidas de seguridad robusta para proteger la información sensible de clientes.
  5. **Soporte y Mantenimiento:** Proporcionar soporte técnico y mantenimiento continuo para asegurar la disponibilidad y funcionalidad del sistema.

Estas características son fundamentales para optimizar las operaciones de una concesionaria o empresa de ventas de automóviles, mejorando la experiencia tanto para los vendedores como para los clientes.

* ¿Qué tipo de datos se deben almacenar y gestionar (cliente, productos, ventas)?
  1. Clientes:}
     + Información personal: Nombres completos, dirección, número de teléfono correo electrónico.
  2. Productos (Automóviles):
     + Detalles del vehículo:
       - Marca, modelo, año, color, número de serie, características.
  3. Venta:
     + Detalle de la venta, fecha de la venta, vendedor asignado, cliente involucrado.
     + Información del vehículo vendido modelo. Precio de venta. Descuento aplicado.
  4. Inventario:
     + Estado actual del inventario: Disponibilidad de vehículos, ubicación en el concesionario.
     + Historial de movimientos: Entradas y salidas de vehículos (vendidos, disponibles).
  5. Finanzas:
     + Información fiscal: Impuestos aplicados a las ventas.

Estos datos son esenciales para proporcionar un servicio personalizado a los clientes, gestionar eficientemente el inventario, optimizar las estrategias de ventas y cumplir con los requisitos regulatorios y fiscales. Un adecuado manejo y almacenamiento de estos datos garantiza un sistema de ventas de vehículos efectivo y bien administrado.

**Interfaz de Usuario:**

¿Cómo debería ser la interfaz de usuario? ¿Hay algún diseño o tema específico en mente?

1. **Paleta de Colores**: Utiliza el azul como color principal para fondos y áreas de enfoque, ya que transmite confianza y profesionalismo. El amarillo se puede emplear como acento para resaltar botones importantes o llamadas a la acción, proporcionando contraste y atención visual. El negro se puede utilizar para textos y elementos de navegación, asegurando legibilidad y claridad.
2. **Diseño Limpio y Funcional**: Opta por un diseño minimalista que facilite la navegación y la interacción intuitiva. Utiliza espacios en blanco para separar elementos y evitar la saturación visual, lo que ayuda a centrar la atención en la información clave.
3. **Elementos Gráficos**: Incorpora iconos y botones claros y distintivos que indiquen funcionalidades específicas.
4. **Diseño Responsivo**: Asegúrate de que la interfaz sea adaptable a diferentes dispositivos y tamaños de pantalla, garantizando una experiencia de usuario consistente y efectiva tanto en computadoras de escritorio como en dispositivos móviles.
5. **Tipografía Clara**: Utiliza fuentes legibles y profesionales para garantizar que la información sea fácil de leer y comprender, especialmente en formularios y detalles de productos.
6. **Accesibilidad**: Considera la accesibilidad al diseñar la interfaz, asegurando que los colores tengan suficiente contraste y que los elementos sean fácilmente accesibles para todos los usuarios.
7. **Intuitiva y fácil de usar**, utilizando un diseño limpio con Bootstrap.

¿Qué páginas y componentes debe tener la interfaz?

Al seguir estas directrices, puedes crear una interfaz de usuario efectiva y atractiva que no solo refleje la identidad visual del sistema de ventas de vehículos, sino que también mejore la experiencia del usuario al interactuar con el sistema.

**Requerimientos Técnicos:**

¿Hay algún requerimiento específico de rendimiento o escalabilidad?

* Soportar hasta 1000 productos y 100 transacciones diarias inicialmente.

**Seguridad**

¿Qué medidas de seguridad deben implementarse?

* Autenticación y autorización de usuarios, cifrado de datos sensibles.

#### **Modelo de Casos de Uso**

**Actores Principales:**

1.Vendedor

2.Cliente

**Casos de Uso:**

1.Gestionar inventario.

2.Procesar ventas.

3.Gestionar usuarios.

**Diagrama de Casos de Uso:**

+------------------+ +------------------+  
| Administrador | | Vendedor |  
+------------------+ +------------------+  
 | |  
 | |  
+--------+--------+ +--------+--------+  
| Gestionar | | Procesar Venta |  
| Inventario | | |  
+--------+--------+ +--------+--------+  
 |   
 |   
 +--------+--------+   
 |   
 |   
 +--------+--------+   
 |   
 |  
 +--------+--------+  
 | Gestionar |  
 | Usuarios |  
 +-----------------+

#### **Modelo de Dominio**

**Entidades Principales:**

1.Producto

2.Cliente

3.Venta

4.Usuario

**Diagrama de Clases:**

### +-------------+ +-------------+ +-------------+ +-------------+ | Producto | | Cliente | | Venta | | Usuario | +-------------+ +-------------+ +-------------+ +-------------+ | -id: int | | -id: int | | -id: int | | -id: int | | -nombre: | | -nombre: | | -productoId:| | -username: | | String | | String | | int | | String | | -precio: | | -email: | | -clienteId: | | -password: | | double | | String | | int | | String | | -stock: int | +-------------+ | -cantidad: | | date | +-------------+ | -fecha | -rol: String| +-------------+

### **1. Pruebas Unitarias en PHP**

Vamos a suponer que tienes una clase CarSales con métodos para añadir y eliminar ventas. Estas pruebas se realizarán utilizando PHPUnit.

**Módulo de añadir Producto**:

<?php include("../../../app/config.php");

if($\_POST){

print\_r($\_POST);

$producto = (isset($\_POST['producto'])? $\_POST['producto']:'');

$cantidad\_inventario = (isset($\_POST['cantidad\_inventario'])? $\_POST['cantidad\_inventario']:'');

$cant\_ingreso\_inven = (isset($\_POST['cant\_ingreso\_inven'])? $\_POST['cant\_ingreso\_inven']:'');

$precio\_unid\_inven = (isset($\_POST['precio\_unid\_inven'])? $\_POST['precio\_unid\_inven']:'');

$precio\_unid\_venta = (isset($\_POST['precio\_unid\_venta'])? $\_POST['precio\_unid\_venta']:'');

$cantidad\_comprar = (isset($\_POST['cantidad\_comprar'])? $\_POST['cantidad\_comprar']:'');

$precio\_total\_venta = (isset($\_POST['precio\_total\_venta'])? $\_POST['precio\_total\_venta']:'');

$precio\_total\_inven = (isset($\_POST['precio\_total\_inven'])? $\_POST['precio\_total\_inven']:'');

$ingreso = (isset($\_POST['ingreso'])? $\_POST['ingreso']:'');

$egreso = (isset($\_POST['egreso'])? $\_POST['egreso']:'');

$fecha = (isset($\_POST['fecha'])? $\_POST['fecha']:'');

$provedor = (isset($\_POST['provedor'])?$\_POST['provedor']:'');

$figura = (isset($\_FILES['figura']['name'])? $\_FILES['figura']['name']: '');

$fecha\_figura = new DateTime();

$nombre\_figura = ($figura)? $fecha\_figura->getTimestamp()." \_ ".$\_FILES['figura']['name']:'';

$tmp\_figura = $\_FILES['figura']['tmp\_name'];

if($tmp\_figura != ''){

move\_uploaded\_file($tmp\_figura,'../../../public/archivos/imgenes/'.$nombre\_figura);

}

$sql = "INSERT INTO tb\_libreria(id,

producto,

cantidad\_inventario,

cant\_ingreso\_inven,

precio\_unid\_inven,

precio\_unid\_venta,

cantidad\_comprar,

precio\_total\_venta,

precio\_total\_inven,

ingreso,

egreso,

fecha,

provedor,

figura)

VALUES(null,:producto,

:cantidad\_inventario,

:cant\_ingreso\_inven,

:precio\_unid\_inven,

:precio\_unid\_venta,

:cantidad\_comprar,

:precio\_total\_venta,

:precio\_total\_inven,

:ingreso,

:egreso,

:fecha,

:provedor,

:figura)";

$sentencia = $pdo->prepare($sql);

$sentencia->bindParam(':producto',$producto);

$sentencia->bindParam(':cantidad\_inventario',$cantidad\_inventario);

$sentencia->bindParam(':cant\_ingreso\_inven',$cant\_ingreso\_inven);

$sentencia->bindParam(':precio\_unid\_inven',$precio\_unid\_inven);

$sentencia->bindParam(':precio\_unid\_venta',$precio\_unid\_venta);

$sentencia->bindParam(':cantidad\_comprar',$cantidad\_comprar);

$sentencia->bindParam(':precio\_total\_venta',$precio\_total\_venta);

$sentencia->bindParam(':precio\_total\_inven',$precio\_total\_inven);

$sentencia->bindParam(':ingreso',$ingreso);

$sentencia->bindParam(':egreso',$egreso);

$sentencia->bindParam(':fecha',$fecha);

$sentencia->bindParam(':provedor',$provedor);

$sentencia->bindParam(':figura',$nombre\_figura);

$sentencia->execute();

header("location:ingresoMain.php");

}

?>

**Módulo de eliminar Producto:**

<?php

include("../../../app/config.php");

/\*verificacion si exite session con var global session start\*/

session\_start();

if(isset($\_SESSION['session\_email'])){

echo "\nsi existe session de ".$\_SESSION['session\_email'];

}else{

echo "\nno existe session";

header('location:../../../index.php');

}

if (isset($\_GET['txtId'])) {

# code...

print\_r($\_GET['txtId']);

$txtId = (isset($\_GET['txtId'])? $\_GET['txtId']: '') ;

$sql = "DELETE FROM tb\_libreria WHERE id=:id";

$sentencia = $pdo->prepare($sql);

$sentencia->bindParam(':id',$txtId);

$sentencia->execute();

header("location:ingresoMain.php");

}

$sentencia = $pdo->prepare("SELECT \* FROM tb\_libreria WHERE 1");

$sentencia->execute();

$ventas = $sentencia->fetchAll(PDO::FETCH\_ASSOC);

?>

<td>

<a

name=""

id=""

class="btn btn-dark"

href="javascript:limpiar(<?php echo $venta['id'];?>)"

role="button"

>

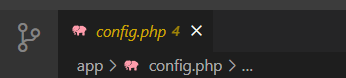
<i class="fa-solid fa-trash text-white"></i>

</

### **2. Pruebas de Integración en PHP**

Las pruebas de integración verificarán cómo interactúan diferentes componentes del sistema, como la base de datos y la clase CarSales.

#### **Pruebas de Integración**



<?php

define('HOST','localhost');

define('USER','root');

define('PASSWORD','');

define('DBNAME','carros');

$servidor = "mysql:dbname=".DBNAME.";host=".HOST;

$fecha = new DateTime();

try {

//code...

$pdo = new PDO($servidor,USER,PASSWORD,array(PDO::MYSQL\_ATTR\_INIT\_COMMAND =>"SET NAMES utf8"));

//echo "la conexion de hoy ".$fecha->format('Y-m-d')." correctamente desde config";

} catch (PDOException $ex) {

//throw $th;

echo "conexion de hoy ".$fecha->format('Y-m-d')." incorrecta algo salio mal en el servidor desde config";

}

?>

3. Validación de Datos en los Inputs:

* En el input para insertar nombres: Solo debe aceptar solo letras y no debe aceptar caracteres como \* -+?¡¿#@, máximo de límite del nombre tres palabras.
* Input Gmail: Como primera palabra acepta palabras minúsculas y mayúsculas numeros excepto caracteres como -seguido esta palabra por una arroba la terminación de la extensión es por gmail.com, gmail.or, gmail.bol,gmail.mx,o por Hotmail.
* Input contraseña: acepta números letras y caracteres, y la longitud mínima de esta debe ser de 9 caracteres.
* Input Nombre del producto: primero es que es el producto y la marca seguido por el año.
* Input Proveedor: Solo acepta letras como mínimo dos palabras.
* Input cantidad Inventario: solo acepta números positivos longitud mínimo 1 máximo 30.
* Input Precio unidad inventario: solo números no acepta caracteres especiales y solo números positivos longitud mínimo 2000 y longitud máxima 20000.
* Input Unidad Venta: solo acepta números no acepta caracteres especiales y numeros positivos no entra números negativos longitud minima del número es 7200 y longitud máxima es de 99000.
* Input Cantidad Comprar: solo acepta números positivos y longitud mínima es de 1 y longitud máxima es de 7.
* Input Nombre cliente: solo acepta letras no acepta caracteres especiales tampoco acepta número, el input requiere de un nombre y dos apellidos.

4. Pruebas de Caja Negra, Caja Blanca y Caja Gris:

#### **Pruebas de Caja Negra**

Las pruebas de caja negra se enfocan en probar la funcionalidad sin conocer la implementación interna. A continuación, algunos ejemplos.

* + Probar con una dirección de email válida e inválida.
  + Probar con precios válidos (positivos) e inválidos (negativos, texto).

#### **Pruebas de Caja Blanca**

Las pruebas de caja blanca se enfocan en probar las estructuras internas del código. Para esto, revisa la lógica de los métodos en la clase CarSales.

* + Verificar que addSale lance excepciones apropiadas para emails y precios inválidos.
  + Verificar que removeSale elimine correctamente una venta dada.

#### **Pruebas de Caja Gris**

Las pruebas de caja gris combinan técnicas de caja blanca y negra. A continuación, algunos ejemplos.

* + Verificar que los emails pasen por la validación de formato antes de insertarse en la base de datos.
  + Verificar que los precios se validen antes de ser añadidos a la base de datos.

Arquitectura del sistema

La arquitectura del sistema se divide en tres capas principales:

1. Capa de Presentación(Frontend):
   1. Esta capa es responsable de la interfaz de usuario y la expereencia del usuario. Utiliza tecnologías como Bootstrap,CSS.Datatables.
      1. Boostrap: Para crear una interfaz de usuario responsive y atractiva.
      2. Css:Para personalizar el estilo y la apariencia.
      3. DataTablas: Para mostrar y gestionar tablas de datos de manera interactiva.
2. Capa de lógica de Negocio(Backend)
3. Capa de Datos (Base de Datos)

Componentes:

HTML/CSS/Bootstrap:

1. Index.html:Pagina de inicio del sistema.
2. Sessiones/login/index.php:Para iniciar session con tu cuenta de usuario ya creado,
3. app/controllers/login/crear.php:Para crear productos en el inventario por primera ves.
4. app/controllers/login/comprar\_editar.php:Para ejecutar una compra.
5. App/controllers/login/add\_product\_inven.php:para el ingreso de productos en el inventario ya de un producto creado.

**JavaScript/DataTables:**

* + **autos.js:** Archivo para manejar la lógica de DataTables en la página de autos.
  + **ventas.js:** Archivo para manejar la lógica de DataTables en la página de ventas.
  + **main.js:** Archivo para manejar la lógica general del frontend.

### **Capa de Datos (Base de Datos)**

Esta capa maneja el almacenamiento y la recuperación de datos. Utiliza MySQL para la base de datos.

**Componentes:**

**Tablas:**

* + **autos:** Almacena la información de los autos.
    - **id (INT):** Identificador único del auto.
    - **marca (VARCHAR):** Marca del auto.
    - **modelo (VARCHAR):** Modelo del auto.
    - **año (INT):** Año del auto.
    - **precio (DECIMAL):** Precio del auto.
  + **ventas:** Almacena la información de las ventas.
    - **id (INT):** Identificador único de la venta.
    - **auto\_id (INT):** Identificador del auto vendido (relación con la tabla autos).
    - **fecha (DATE):** Fecha de la venta.
    - **cantidad (INT):** Cantidad de autos vendidos.
    - **total (DECIMAL):** Total de la venta.

1. **Conexión a la Base de Datos:**
   * **db.php:** Archivo que maneja la conexión a la base de datos.

