TEMA DE INVESTIGACIÓN:

MANUAL TÉCNICO EXPLICANDO EL USO DE SITIO WEB PARA PROYECTO TÉCNICO CIENTÍFICO, SISTEMA DARG.

PRESENTADO POR:

ADRIANA PAOLA MEJÍA MÉNDEZ, MELANIE JACKELINE MARTÍNEZ RAMÍREZ, EMILY GUADALUPE MURILLO ARGUETA, DANIEL ALEJANDRO CORTEZ QUINTANILLA, AXEL GABRIEL GARCÍA RAMÍREZ.

TERCER AÑO DE BACHILLERATO,

ESPECIALIDAD DE DESARROLLO DE SOFTWARE

INSTITUTO TÉCNICO RICALDONE

SAN SALVADOR, OCTUBRE 2024

Índice

1. Introducción

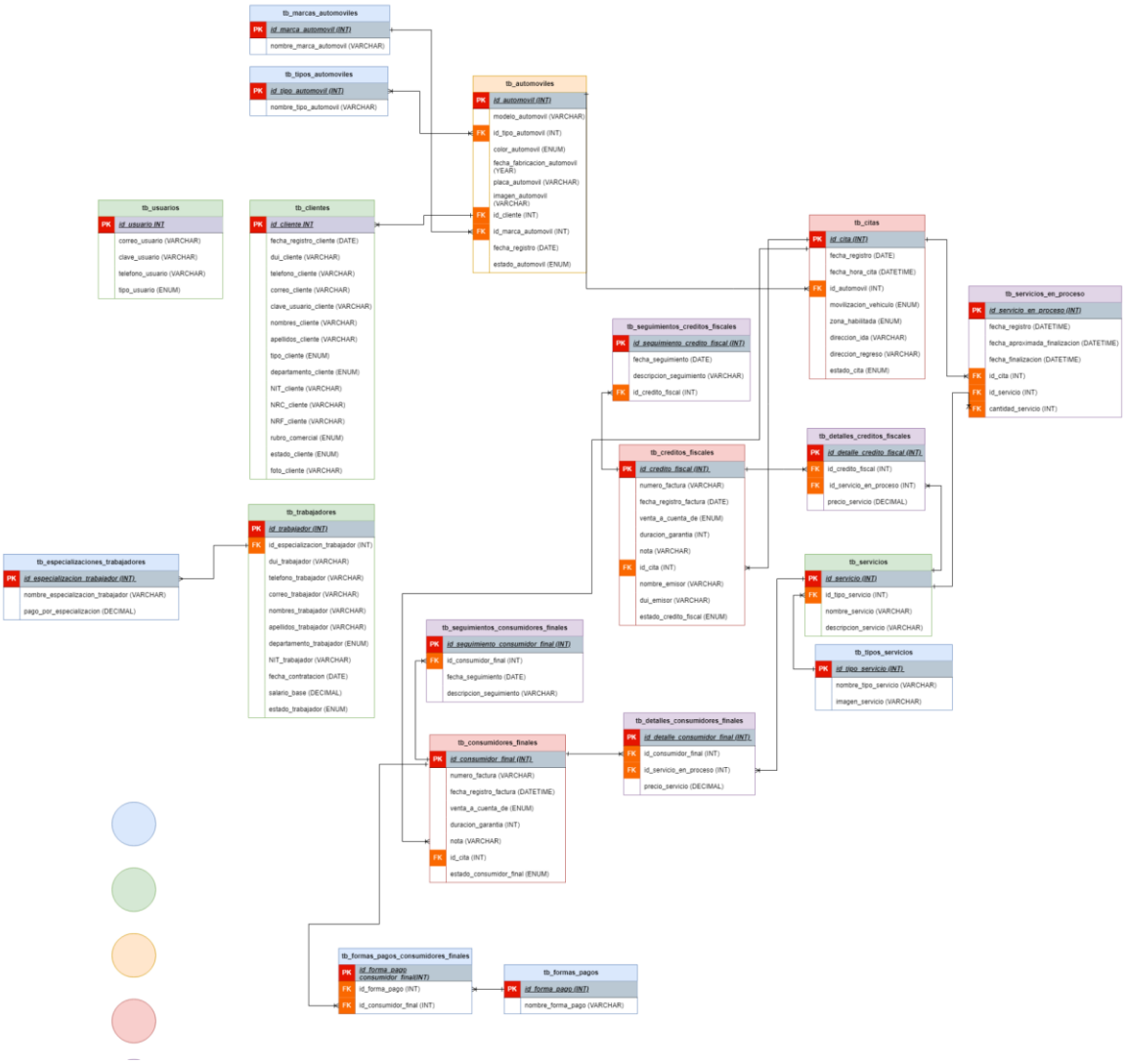
El presente manual técnico del sitio web “Data Administration Revolution Garage” (DARG) tiene como propósito ofrecer una guía completa y detallada para facilitar la comprensión, el uso y el mantenimiento efectivo del sitio web privado. Este manual está diseñado especialmente para el usuario de nivel “Administrador”, quien será el encargado de operar el sistema, así como para los desarrolladores, como un recurso útil de referencia. A lo largo de este documento, se abordarán los siguientes apartados:

1. **Tecnologías y herramientas utilizadas**: Se describen los recursos empleados en el desarrollo del sistema web.
2. **Estructura de la base de datos**: Se presenta el modelo relacional de la base de datos con una imagen y el código SQL correspondiente.
3. **Diccionario de datos**: Se define cada elemento de la base de datos en un formato estandarizado.
4. **Arquitectura del software**: Se detallan las capas de la aplicación web y su interacción, incluyendo diagramas ilustrativos.
5. **Estructura del proyecto**: Se aborda la organización de directorios y archivos que componen la aplicación web y su relación.
6. **Diseño de la aplicación**: Se describen las características visuales utilizadas.
7. **Buenas prácticas de desarrollo**: Se establecen los estándares de programación.
8. **Requerimientos de hardware y software**: Se detallan las condiciones mínimas y óptimas necesarias para el funcionamiento del sistema.
9. **Instalación y configuración:** Se ofrecen instrucciones para la puesta en marcha del sistema, tanto local como en línea.

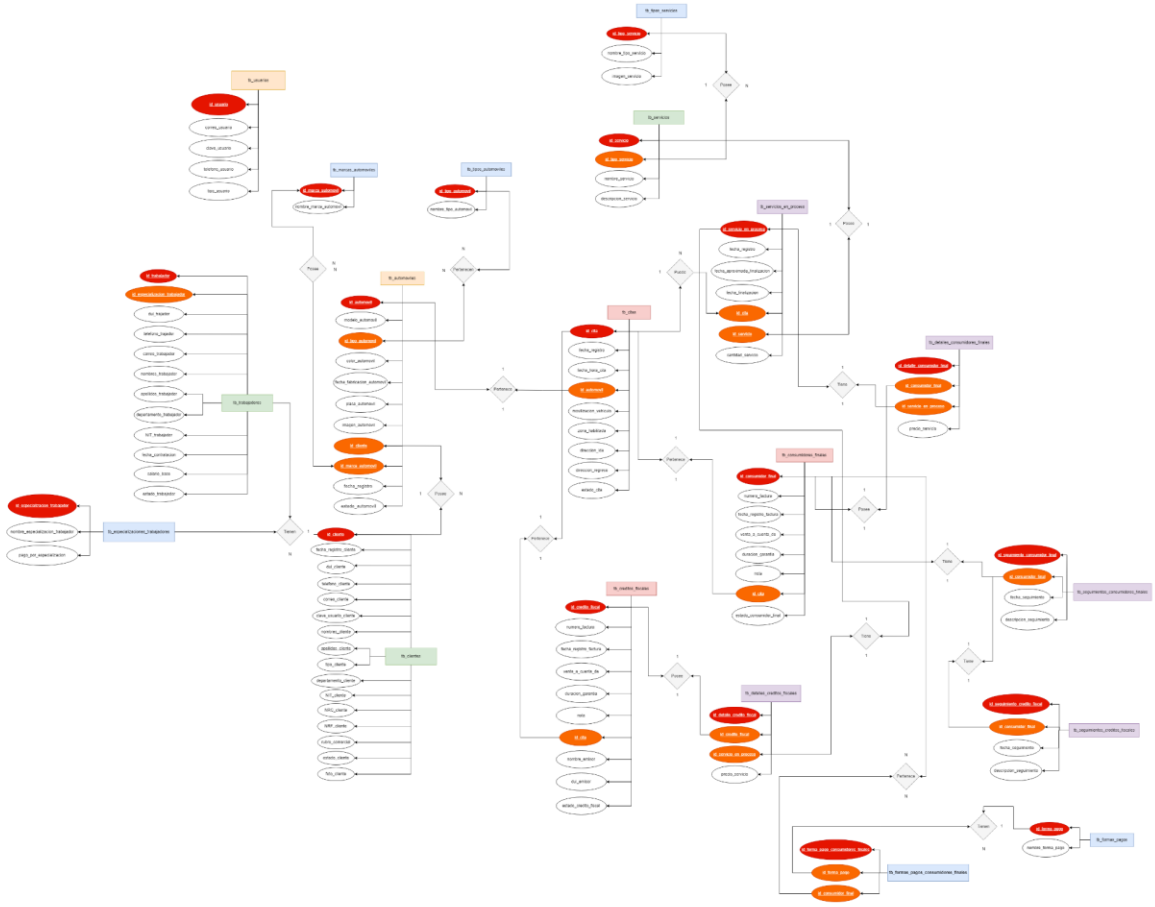
2. Tecnologías y herramientas empleadas.

- **HTML5:** Lenguaje base para la estructura de la página, define elementos visibles como botones, formularios e imágenes.
- **CSS:** Lenguaje para estilizar la página, permitiendo personalizar colores, fuentes y diseños.
- **JAVASCRIPT:** Lenguaje que añade interactividad y dinamismo, permitiendo validaciones y operaciones en segundo plano.
- **PHP:** Lenguaje servidor que conecta la base de datos con las vistas, gestionando consultas y seguridad.
- **MYSQL:** Sistema de gestión de bases de datos relacional para almacenar información de usuarios y funcionalidades.
- **JQUERY:** Librería que facilita la manipulación del DOM y mejora la búsqueda de datos ingresados.
- **APACHE:** Servidor web para implementar la aplicación tanto localmente como en Google Cloud.
- **BOOTSTRAP:** Framework que simplifica el diseño de componentes responsivos y atractivos, acelerando el desarrollo.
- **GOOGLE CLOUD:** Plataforma que aloja el sitio web y conecta aplicaciones, asegurando su funcionamiento correcto.
- **DRAW.IO:** Herramienta para crear diagramas de la estructura de la base de datos, proporcionando una visión clara del modelo de datos.
- **FIGMA:** Herramienta de diseño colaborativo para crear interfaces atractivas y funcionales tanto para web como para móvil.
- **LIBRERÍAS:** Las librerías utilizadas dentro del proyecto del sitio web fueron phpmailer que se utiliza para el servicio de envío de correos electrónicos a los usuarios con propósitos de cambios de contraseña y la librería fpdf que se utiliza en el apartado de reportes.
    - **PHPMailer:** Para el envío de correos electrónicos.
    - **FPDF:** Para generación de reportes en PDF.

3. Estructura de la base de datos



Link diagrama modelo de dominio: [diagrama_modelodedominio_revolutiongarage.drawio]



Link diagrama entidad relación: [diagrama_entidadrelacion_revolutiongarage.drawio]

Script de la base de datos (sin datos insertados): [darg_database_script.sql]

4. Diccionario de datos (siguiente página):

tb_clientes					
Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_cliente	INT		NO	PRIMARY KEY	AUTONÚMERICO
fecha_registro_cliente	DATE		NO		CURRENT_TIMESTAMP
dui_cliente	VARCHAR	10	NO	UNIQUE	
telefono_cliente	VARCHAR	9	NO	UNIQUE	
correo_cliente	VARCHAR	50	NO	UNIQUE	
clave_cliente	VARCHAR	100	NO	DEFAULT	00000000
nombres_cliente	VARCHAR	50	NO		
apellidos_cliente	VARCHAR	50	NO		
tipo_cliente	ENUM		NO		
departamento_cliente	ENUM		NO		
NIT_cliente	VARCHAR	18	SÍ	UNIQUE	
NRC_cliente	VARCHAR	11	SÍ	UNIQUE	
NRF_cliente	VARCHAR	25	SÍ	UNIQUE	
rubro_comercial	ENUM		SÍ		
estado_cliente	ENUM		NO	DEFAULT	TRUE
foto_cliente	VARCHAR	50	SÍ		

tb_especializaciones_trabajadores					
Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_especializacion_trabajador	INT		NO	PRIMARY KEY	AUTONÚMERICO
nombre_especializacion_trabajador	VARCHAR	100	NO		
pago_por_especializacion	DECIMAL		NO		

tb_trabajadores					
Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_trabajador	INT		NO	PRIMARY KEY	AUTONÚMERICO
id_especializacion_trabajador	INT		NO	FOREING KEY	
dui_trabajador	VARCHAR	10	NO	UNIQUE	
telefono_trabajador	VARCHAR	9	NO	UNIQUE	
correo_trabajador	VARCHAR	50	NO	UNIQUE	
nombres_trabajador	VARCHAR	50	NO		
apellidos_trabajador	VARCHAR	50	NO		
departamento_trabajador	ENUM		NO		
NIT_trabajador	VARCHAR	18	SÍ	UNIQUE	
fecha_contratacion	DATE		NO		CURRENT_DATE
salario_base	DECIMAL	5.2	NO		

tb_marcas_automoviles					
Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_marca_automovil	INT		NO	PRIMARY KEY	AUTONÚMERICO
nombre_marca_automovil	VARCHAR	50	NO		

tb_tipos_automoviles					
Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_tipo_automovil	INT		NO	PRIMARY KEY	AUTONÚMERICO
nombre_tipo_automovil	VARCHAR	80	NO		

tb_automoviles					
Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_automovil	INT		NO	PRIMARY KEY	AUTONÚMERICO
modelo_automovil	VARCHAR	50	NO		
id_tipo_automovil	INT		NO	FOREING KEY	
color_automovil	ENUM		NO		
fecha_fabricacion_automovil	YEAR		NO		
placa_automovil	VARCHAR	11	NO	UNIQUE	
imagen_automovil	VARCHAR	25	NO		
id_cliente	INT		NO	FOREING KEY	
id_marca_automovil	INT		NO	FOREING KEY	
fecha_registro	DATE		NO		CURRENT_DATE
estado_automovil	ENUM		NO		´ACTIVO´

tb_tipos_servicios					
Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_tipo_servicio	INT		NO	PRIMARY KEY	AUTONÚMERICO
nombre_tipo_servicio	VARCHAR	50	NO		
imagen_servicio	VARCHAR	25	NO		

tb_servicios					
Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_servicio	INT		NO	PRIMARY KEY	AUTONÚMERICO
id_tipo_servicio	INT		NO	FOREING KEY	
nombre_servicio	VARCHAR	50	NO		
descripcion_servicio	VARCHAR	50	NO		

tb_formas_pagos					
Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_forma_pago	INT		NO	PRIMARY KEY	AUTONÚMERICO
nombre_forma_pago	VARCHAR	100	NO		

tb_citas					
Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_cita	INT		NO	PRIMARY KEY	AUTONÚMERICO
fecha_registro	DATE		NO		
fecha_hora_cita	DATETIME		NO		
id_automovil	INT		NO	FOREING KEY	
movilizacion_vehiculo	ENUM		NO		
zona_habilitada	ENUM		SÍ	UNIQUE	
direccion_ida	VARCHAR	250	SÍ		
direccion_regreso	VARCHAR	250	SÍ	FOREING KEY	
estado_cita	ENUM		NO		‘EN ESPERA’

tb_servicios_en_proceso					
Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_servicio_en_proceso	INT		NO	PRIMARY KEY	AUTONÚMERICO
fecha_registro	DATETIME		NO		
fecha_aproximada_finalizacion	DATETIME		NO		
fecha_finalizacion	DATETIME		SÍ		
id_cita	INT		NO	FOREING KEY	
id_servicio	INT		NO	FOREING KEY	
cantidad_servicio	INT		NO		

tb_consumidores_finales					
Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_consumidor_final	INT		NO	PRIMARY KEY	AUTONÚMERICO
numero_factura	VARCHAR	5	NO		
fecha_registro_factura	DATETIME		NO	DEFAULT	CURRENT_DATE
venta_a_cuenta_de	ENUM		SÍ		
duracion_garantia	INT		NO	FOREING KEY	
nota	VARCHAR	100	SÍ	FOREING KEY	
id_cita	INT		NO		
estado_consumidor_final	ENUM		NO	DEFAULT	‘EN ESPERA’

tb_detalles_consumidores_finales					
Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_detalle_consumidor_final	INT		NO	PRIMARY KEY	AUTONÚMERICO
id_consumidor_final	INT		NO	FOREING KEY	
id_servicio_en_proceso	INT		NO	FOREING KEY	
precio_servicio	DECIMAL		NO	DECIMAL	

tb_creditos_fiscales					
Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_credito_fiscal	INT		NO	PRIMARY KEY	AUTONÚMERICO
numero_factura	VARCHAR	5	NO		
fecha_registro_factura	DATE		NO	DEFAULT	
venta_a_cuenta_de	ENUM		NO		
duracion_garantia	INT		NO		
nota	VARCHAR	100	SÍ		
id_cita	INT		NO	FOREING KEY	
nombre_emisor	VARCHAR	25	SÍ		
dui_emisor	VARCHAR	10	NO	UNIQUE	
estado_credito_fiscal	ENUM		NO		

tb_detalles_creditos_fiscales

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_detalle_credito_fiscal	INT		NO	PRIMARY KEY	AUTONÚMERICO
id_credito_fiscal	INT		NO	FOREING KEY	
id_servicio_en_proceso	INT		NO	FOREING KEY	
precio_servicio	DECIMAL		NO	DECIMAL	

tb_seguimientos_consumidores_finales

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_seguimiento_consumidor_final	INT		NO	PRIMARY KEY	AUTONÚMERICO
id_consumidor_final	INT		NO		
fecha_seguimiento	DATE		NO		CURRENT_DATE
descripcion_seguimiento	VARCHAR	500	NO		

tb_seguimientos_creditos_fiscales

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_seguimiento_credito_fiscal	INT		NO	PRIMARY KEY	AUTONÚMERICO
fecha_seguimiento	DATE		NO		CURRENT_DATE
descripcion_seguimiento	VARCHAR	500	NO		
id_consumidor_final	INT		NO		

tb_usuarios					
Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_usuario	INT		NO	PRIMARY KEY	AUTONÚMERICO
correo_usuario	VARCHAR	200	NO		
clave_usuario	VARCHAR	100			
telefono_usuario	VARCHAR	9	NO		
tipo_usuario	ENUM				

tb_formas_pagos_consumidores_finales					
Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_forma_pago_consumidores_finales	INT		NO	PRIMARY KEY	AUTONÚMERICO
id_forma_pago	INT	50	NO		
id_consumidor_final	INT	25	NO		

5. Arquitectura del software

- **Backend:** Gestiona toda la lógica de negocio, la conexión a la base de datos y la construcción de la API, facilitando el intercambio de información con las solicitudes del usuario. Asegura una interacción fluida entre el frontend y el servidor, garantizando que la aplicación funcione correctamente y entregue los datos solicitados de manera eficiente. Las tareas del backend se organizan en distintas carpetas, como una carpeta API que agrupa subcarpetas como helpers (para gestionar la conexión a la base de datos), models (para el manejo de datos y controladores), y services (encargados del intercambio y procesamiento de peticiones).

- **Solicitudes HTTP:** Las solicitudes HTTP permiten el intercambio de información mediante mensajes que el navegador envía al servidor web, facilitando la interacción entre ambos. A través de estas solicitudes, los clientes pueden obtener o enviar datos, haciendo posible la comunicación en la web. Estas solicitudes son esenciales para que el cliente (el navegador) y el servidor intercambien información de manera eficiente y estructurada.

- **Frontend:** Es la parte del software con la que el usuario interactúa directamente. Su objetivo es proporcionar una interfaz amigable para el usuario y, al mismo tiempo, conectarse eficientemente con el backend. En este manual, describimos un enfoque simple usando HTML, CSS y JavaScript puro, sin el uso de framework complejos como React o Angular.

    - Estructura básica del Frontend

    El frontend se organiza principalmente en tres tecnologías:

    - HTML (HyperText Markup Language): Define la estructura del contenido.
    - CSS (Cascading Style Sheets): Controla la apariencia y el diseño visual del contenido.
    - JavaScript: Añade interactividad, como el manejo de eventos y la comunicación con el backend.

      Este enfoque asegura que el sistema mantenga una estructura simple y directa, sin la necesidad de dependencias externas o configuraciones complicadas.

    - Comunicación Frontend-Backend

      Para que el frontend interactúe con el backend, se utilizan solicitudes HTTP. Mediante estas solicitudes, el frontend puede:

    - Obtener datos del backend (GET).
    - Enviar datos al backend (POST).
    - Actualizar información existente (PUT).
    - Eliminar información (DELETE).

6. Estructura del proyecto.

- Carpeta principal: En esta carpeta se encuentran todos los archivos necesarios para el funcionamiento del proyecto con el plan de negocio cliente-servidor con el nombre PROYECTO-TÉCNICO-CIENTÍFICO-2024.
    - API:
        - Helpers: En esta carpeta se encuentran los archivos php que corresponden a la plantilla de factura, las credenciales para la conexión a la base de datos, preparación de las consultas sql con su manejo de errores, plantilla de reportes y validado para cada tipo de dato antes de agregarlo a la consulta para la petición al servidor.
        - Images: En esta carpeta como su nombre lo dice almacena las imágenes que se vayan agregando dentro del programa.
        - Automóviles: Esta carpeta almacena las imágenes de los autos agregados tanto por la aplicación web como por la aplicación móvil, también cuenta con una imagen predeterminada si no se ingresa ninguna en específico.
        - Clientes: Esta carpeta almacena las fotos de perfil de los clientes.
        - Report: Esta carpeta almacena las imágenes que aparecerán en la plantilla de los reportes.
        - Tipo servicio: En esta carpeta se guardarán las imágenes que se agreguen desde la sección de tipo servicios en la aplicación web, igualmente cuenta con una imagen predeterminada en el caso que no se agregue ninguna imagen para el servicio.
            - default.png: (preguntar)
        - Libraries: En esta carpeta se encuentran las diferentes librerías que se utilizaron para el funcionamiento de la aplicación web.
        - fpdf185: Esta librería se utiliza para la creación de las plantillas de los reportes y así mostrar los diferentes tipos de datos que se encuentran en cada uno de los reportes.
        - phpmailer651: Esta librería se utiliza para el envío de correos personalizados para los usuarios, nosotros los usamos con fines de envío de códigos para el cambio de credenciales y verificación de dos pasos.

- Models: En esta carpeta se encuentran los archivos correspondientes al data y al handler que son capas indispensables para el modelo cliente-servidor donde se han los set con sus validaciones y se hacen las consultas a la base de datos.
- Data: En esta carpeta se hacen los set a los valores con sus validaciones correspondientes dependiendo del dato contando con su validador específico que se encuentra en el archivo validator en la carpeta helpers.
- Handler: En esta carpeta se hacen las funciones con las consultas a la base de datos asignando los valores a las mismas anteriormente validados en los archivos data.
- Reports: En esta carpeta se encuentran los archivos php donde se crean los reportes con la plantilla definida en report.php en la carpeta de helpers y se le agregan los datos a partir de consultas a la base de datos dependiendo del reporte.
- Administrador: Aquí se encuentran los reportes que se encuentran en la vista de administrador de la aplicación web cada uno cuenta con una o más consultas dependiendo de cada tipo de reporte, estas consultas se encuentran en la carpeta handler y luego se llaman dentro de estos archivos de reporte.
- Cliente: Igualmente en esta carpeta se almacenan reportes, pero con la diferencia que son para los clientes.
- Services: En esta carpeta de services se encuentran los archivos php donde se envían las consultas a la base de datos con solicitudes Get o Post dependiendo del caso, algunas acciones estrictamente tienen que realizarse con una sesión iniciada para así evitar exponer datos sensibles a cualquier usuario.
- Privado: Aquí se encuentran los envíos de consultas por parte del sitio web privado para realizar las diferentes acciones administrativas y de seguimiento dentro del proyecto.
- Público: En el apartado de público se encuentran los envíos de consultas que se utilizan en la aplicación móvil y estos se encuentran dentro de la misma API para así tener conectividad entre ambas aplicaciones.
- mandar_correo.php: Este archivo se encuentra fuera de ambas carpetas por el hecho de que se utiliza tanto en la aplicación móvil como en la

aplicación web. Este archivo funciona en conjunto con la libreria fpdf185 para el envío de correos con códigos de verificación para seguridad de dos pasos y cambios de contraseña.

- Controlador: Esta carpeta tiene los controladores de cada vista dentro del sitio web, estos se encargan de enviar los campos de los formularios a la API, hacer validaciones y darles un formato a los diferentes campos dentro del sitio web.
- Privado: Aquí se encuentran los controladores del sitio web privado que cumplen con las funcionalidades anteriormente mencionadas siendo cada archivo dependiente de su vista.
- Utilidades: En esta carpeta se encuentran archivos bastante importantes para los controladores ya que estos dependen de el archivo components.js donde se agrega la url del proyecto que luego se complementa por parte de la API dependiendo de la acción que se vaya a realizar, contiene validaciones, inicia la vista de los gráficos, alertas, etc. Se tiene también el control de inactividad y la plantilla de las vistas.

- Recursos: Aquí se encuentran archivos, imágenes, hojas de estilo y manuales de sitio web.
  - Errores: En esta carpeta se encuentran las vistas de los errores 403 que es de recursos restringidos y el error 404 que es error de página o dirección url inexistente.
  - Estilos: Aquí se encuentran las hojas de estilos que se aplican a las vistas del sitio web privado y el archivo de fuentes utilizadas dentro del sitio.
  - Privado: Aquí se encuentran las hojas de estilo que se utilizan en las vistas del sitio web privado incluyendo vista de errores.
  - fuentes.css: Aquí se encuentran las fuentes utilizadas dentro del proyecto.
  - Imágenes: Aquí se encuentran imágenes, iconos y recursos visuales que están dentro del sitio web y son independientes de las imágenes que se encuentran en la carpeta de api quiere decir que estas no cambian.
  - Icons: Como la carpeta lo dice son iconos que se utilizan dentro del sitio que son iconos para controles interactivos e iconos para las pestañas de las páginas.

- img_automoviles: Recursos visuales exclusivos para la pagina de automóviles.
- img_citas: Recursos visuales exclusivos para la vista de citas.
- img_cliente: Recursos visuales exclusivos para la vista de detalle_clientes.
- img_clientes: Recursos visuales exclusivos para la vista de clientes.
- img_empleados: Recursos visuales exclusivos para la vista de empleados.
- img_factura_persona_natural: Recursos visuales exclusivos para la vista de factura_persona_natural.
- img_panel: Recursos visuales exclusivos para la vista de panel_principal.
- img_servicios: Recursos visuales exclusivos para la vista de servicios.
- img-index: Recursos visuales exclusivos para la vista de index.
- img-usuario: Recursos visuales exclusivos para la vista de usuario.
- logos: Recursos visuales relacionados a los logos de la empresa.
- Image403.svg: Recurso visual que se utiliza en la vista de error 403.
- Image404.svg: Recurso visual que se utiliza en la vista de error 404.
- user_exmpl.png: Recurso de visual de ejemplo de foto de perfil masculino.
- user_exmpl2.png: Recurso de visual de ejemplo de foto de perfil femenino.

  - Js: Archivos js utilizados en el panel principal.
  - Manuales: Carpeta que contiene el manual de usuario y el manual técnico.

- Vistas: En esta carpeta se encuentran las vistas en otras palabras las páginas web que están dentro del sitio y que contienen todos los elementos necesarios para realizar las acciones necesarias para cumplir el propósito del proyecto.

        - Privado: En esta carpeta se encuentran las vistas que componen el sitio web privado.
        - Público: En esta carpeta se encuentran las vistas que componen el sitio web público.
    - Otros: En el apartado de otros se encuentran el script de la base de datos utilizada en el proyecto, los diagramas entidad relación y de modelo de dominio, el diccionario de datos y triggers.

7. Diseño de la aplicación

a. Paleta de colores.

- **#FFFFFF (Blanco):** Color de fondo principal, ofrece claridad y espacio.
- **#F5F3F4 (Gris claro):** Utilizado para áreas secundarias o de fondo, añade profundidad sin distraer.
- **#D3D3D3 (Gris medio):** Ideal para bordes, separadores y elementos de menor prioridad.
- **#B1A7A6 (Gris oscuro):** Usado para texto secundario o deshabilitado.
- **#E5383B (Rojo brillante):** Color de fondo principal
- **#E00D11 (Rojo intenso):** Para áreas secundarias en el proyecto.
- **#BA181B (Rojo oscuro):** Utilizado en elementos que requieren énfasis o urgencia.
- **#A4161A (Rojo vino):** Para detalles visuales y gráficos que requieran un tono más sutil.
- **#660708 (Rojo oscuro profundo):** Usado en fondos o elementos que necesitan un contraste fuerte.
- **#161A1D (Negro suave):** Ideal para texto principal, proporciona legibilidad.
- **#0B090A (Negro intenso):** Para elementos que requieren máxima atención.
- **#151313 (Negro carbón):** Se utiliza en elementos de fondo o texto que requieren un enfoque adicional.

Está paleta de colores se ha empleado en diversas partes dentro del sistema, aquí van algunos ejemplos de pantallas que contienen estos colores:

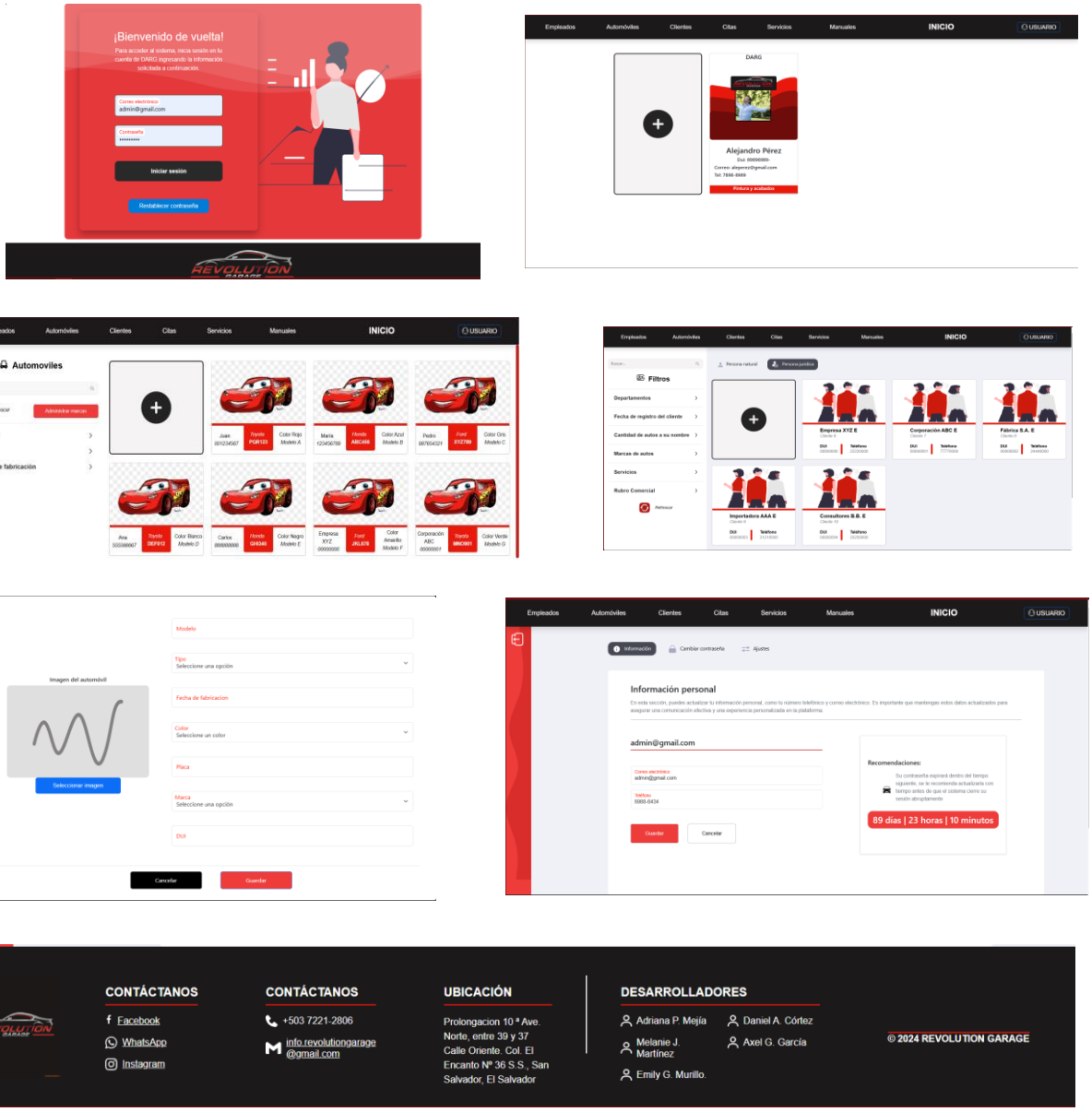
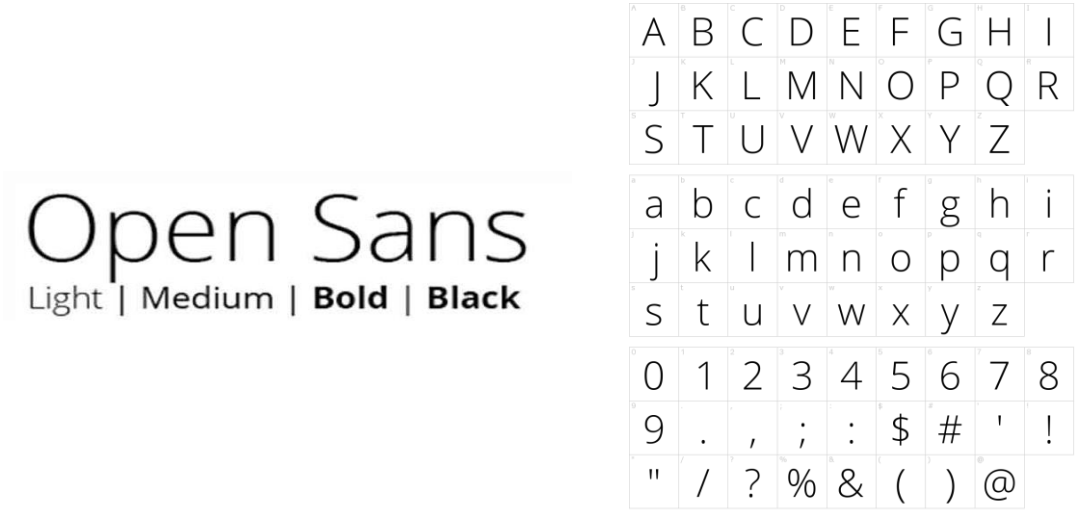


Ilustración 1. Imágenes de ejemplo aplicando la paleta de colores.

b. Tipografía: Open Sans



En nuestro proyecto, utilizamos las etiquetas **h3, h4, h5, h6** y **p** para establecer una jerarquía visual clara:

- **h3**: 22px (títulos principales o secciones importantes).
- **h4**: 18px (subtítulos o encabezados secundarios).
- **h5**: 16px (títulos más pequeños o subsecciones).
- **h6**: 14px (encabezados de menor importancia).
- **p**: 14px (texto de párrafo).

Esta jerarquía de tamaños de fuente asegura que el contenido sea fácilmente legible y que los usuarios puedan identificar de manera rápida la importancia y la estructura del texto.

c. Dimensiones de imágenes.

Las imágenes que se han utilizado a lo largo del proyecto tienen diversos formatos, como tipo SVG y PNG, existen de diferentes tamaños, además de tener diversos momentos de uso, como imágenes ilustrativas, iconos y símbolos de acciones.

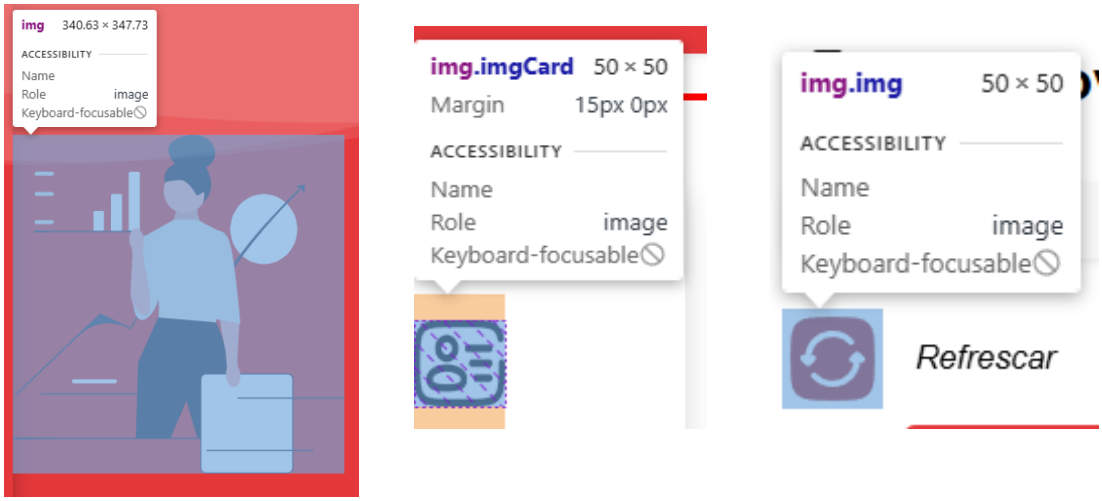


Ilustración 2.Dimensiones de imágenes dentro del sistema.

8. Tamaños soportados en diferentes dispositivos.

Bootstrap es un framework CSS popular que facilita el desarrollo de sitios web **responsivos**, es decir, sitios que se adaptan automáticamente a diferentes tamaños de pantalla para proporcionar la mejor experiencia de usuario, sin importar el dispositivo.

En **Bootstrap V5.3**, los tamaños de pantalla se dividen en seis categorías principales, lo que permite al desarrollador definir estilos y comportamientos específicos para cada tipo de dispositivo:

- **xs (extra small)**:
    - Comprende dispositivos con un ancho de pantalla menor a **576px**.
    - Se trata de dispositivos muy pequeños, como teléfonos móviles de gama baja o viejos smartphones.
    - En estos dispositivos, los elementos suelen ocupar todo el ancho de la pantalla (diseño de una columna).
- **sm (small)**:
    - Comprende dispositivos con un ancho de pantalla de **≥576px**.
    - Incluye la mayoría de los smartphones actuales.
    - Bootstrap permite ajustar el diseño para que los elementos empiecen a mostrar cierta separación o disposición en varias columnas.
- **md (medium)**:
    - Comprende dispositivos con un ancho de pantalla de **≥768px**.
    - Esta categoría incluye tablets y algunos dispositivos móviles grandes.
    - Aquí, el diseño se suele organizar en más columnas, aprovechando el espacio extra disponible.
- **lg (large)**:
    - Comprende dispositivos con un ancho de pantalla de **≥992px**.
    - Incluye la mayoría de las laptops y pantallas de escritorio pequeñas.
    - En este punto, el diseño puede aprovechar aún más el espacio, dividiendo el contenido en múltiples columnas y utilizando imágenes de mayor resolución.
- **xl (extra large)**:
    - Comprende dispositivos con un ancho de pantalla de **≥1200px**.

    - Se refiere principalmente a pantallas de escritorio grandes.
    - Bootstrap ofrece soporte para diseños más complejos, con múltiples columnas y contenido detallado, aprovechando el amplio espacio.
- **xxl (extra extra large)**:
    - Comprende dispositivos con un ancho de pantalla de **≥1400px**.
    - Esta categoría cubre pantallas de muy alta resolución, como monitores de escritorio grandes o pantallas 4K.
    - Aquí, Bootstrap permite crear interfaces más extensas y ricas en contenido, distribuyendo los elementos de manera que se aproveche todo el espacio disponible.

9. Buenas prácticas de desarrollo.

Estándares y buenas prácticas de programación

a. Lado del cliente (JS):

01. **Nomenclatura de Variables y Funciones:**

- Variables: Se utiliza SNAKE_CASE en mayúsculas para nombrar constantes y configuraciones.
- Funciones: Se emplea el formato camelCase para nombrar funciones, para su identificación como acciones o comportamientos.

```
const INFO_PERSONAL = document.getElementById('infoPersonal'); // Constante del contenedor de la información personal del usuario
const CHANGE_CONTRA = document.getElementById('changeContra');  // Constante del contenedor para cambiar la contraseña
const AJUSTES = document.getElementById('ajustes');  // Constante del contenedor para los ajustes

// Función para mostrar el div de la informacion personal y ocultar el de ajustes y cambio de contraseña
function showInfoPersonal(boton) {
    AJUSTES.classList.add('d-none');
    CHANGE_CONTRA.classList.add('d-none');
    INFO_PERSONAL.classList.remove('d-none');
    updateButtonColors(boton);
}
```

02. **Manejo de Errores:**

- Se implementan alertas a través de SweetAlert para proporcionar retroalimentación clara y visual al usuario en caso de éxito o error en las operaciones.

```
// Se comprueba si la respuesta es satisfactoria, de lo contrario se muestra un mensaje con la excepción.
if (DATA.status) {
  handleSuccess(action, form);
} else {
  handleError(DATA);
}
```

03. **Validaciones Previas:**

- Antes de enviar formularios o realizar operaciones críticas , se llevan a cabo validaciones exhaustivas de las entradas del usuario. Además, se muestran mensajes de confirmación para asegurar que el usuario esté consciente de las acciones que está a punto de realizar.

```
const IS_VALID = await checkFormValidity(form);
if (IS_VALID) { //Si retorna true se ejcuta el codigo
```

```
let TITLE, MESSAGE;
if (estado_cita === 'Cancelado') {
  TITLE = '¿Seguro que quieres cancelar la cita?';
  MESSAGE = 'Una vez cancelada solo se podrá eliminar';
} else if (estado_cita === 'Aceptado') {
  TITLE = '¿Seguro que quieres aceptar la cita?';
  MESSAGE = 'Una vez aceptada no podrás agendar citas con la misma fecha y hora';
} else {
  TITLE = '¿Seguro que quieres realizar esta acción?';
  MESSAGE = '';
}

// Llamada a la función para mostrar un mensaje de confirmación, capturando la respuesta en una constante.
const RESPONSE = await confirmAction2(TITLE, MESSAGE);
```

04. **Funciones Reutilizables:**

- Se estructuran funciones específicas para tareas recurrentes, como cargar datos, mostrar imágenes y realizar confirmaciones de acción.

```
//Funcion para mostrar los distintos errores que podrian aparecer
const handleError = async (DATA) => {
  if (DATA.error === 'Acción no disponible fuera de la sesión, debe ingresar para continuar') {
    await sweetAlert(4, DATA.error, true);
    location.href = 'index.html';
  } else {
    sweetAlert(4, DATA.error, true);
  }
};
```

b. Lado del servidor (PHP):

01. **Uso de Clases y Métodos**

- El código está organizado en clases, lo que representa una buena práctica dentro de la programación orientada a objetos. Esta estructura permite encapsular la funcionalidad relacionada y mantener el código más limpio y modular.

02. **Nomenclatura de Clases, Variables y Funciones**

- Clases: Se utiliza el formato UpperCamelCase para nombrar las clases, lo que es estándar en la mayoría de los lenguajes de programación orientados a objetos.

```
class UsuariosClientesHandler
```

- Variables: Se emplea el formato snake_case para nombrar las variables, lo que mejora la legibilidad y facilita la identificación de los elementos.

```php
//Declaracion de atributos para el manejo de los datos de la tabla en la base de datos
protected $id_usuario_cliente = null;
protected $clave_usuario_cliente = null;
protected $estado_usuario = null;
protected $id_cliente = null;
protected $correo_usuario = null;
protected $dui_cliente = null;
protected $telefono_cliente = null;
```

- Funciones: Se utiliza el formato camelCase para nombrar las funciones, permitiendo distinguir claramente estas acciones o comportamientos dentro del código.

```php
//Funcion que valida el correo
public function checkCorreo()
```

03. **Propiedades de Clase**

- Las propiedades de la clase están declaradas como protected, lo que permite el acceso a ellas desde la misma clase y las clases que la heredan, mientras que se oculta su acceso desde fuera.

```php
//Declaracion de atributos para el manejo
protected $id_usuario_cliente = null;
protected $clave_usuario_cliente = null;
```

04. **Preparación de Consultas SQL**

- Las consultas SQL utilizan parámetros en lugar de concatenar directamente las variables, lo que ayuda a prevenir inyecciones SQL.

```php
$sql = 'SELECT cl.* FROM tb_clientes cl WHERE cl.correo_cliente = ?;'; // Consulta SQL para verificar un usuario con ese correo
$params = array($this->correo_usuario); // Parámetros para la consulta SQL
return Database::getRow($sql, $params); // Ejecución de la consulta SQL
```

05. **Consistencia en Nombres de Métodos**

- Los nombres de los métodos son descriptivos y reflejan su propósito (por ejemplo, readAll, createRow, deleteRow).

```php
public function readProfile()
```

```php
public function createRowPersonaNatural()
```

```
public function createRowPersonaJuridica()
```

c. Buenas prácticas para la base de datos

1. Nomenclatura de Tablas, Campos y sentencias SQL

- Tablas: Los nombres de las tablas son descriptivos y reflejan claramente el contenido o la función de cada una. Se utilizan en formato plural y se acompañan con el prefijo tb_, siguiendo el estilo snake_case.

```
tb_clientes
```

- Campos: Todos los nombres de los campos utilizan el estilo snake_case, son descriptivos y reflejan claramente el contenido o la función que desempeñan dentro de la tabla, además de incluir el sufijo _tabla para indicar de manera explícita la pertenencia a la entidad de esa tabla.

```
id_cliente INT AUTO_I
fecha_registro_client
dui_cliente VARCHAR(1
telefono_cliente VARC
correo_cliente VARCHA
clave_usuario_cliente
nombres_cliente VARCH
apellidos_cliente VAR
```

- Sentencias SQL: Las sentencias SQL se escriben en mayúsculas para diferenciarlas claramente del resto del código, lo que facilita su identificación y lectura.

```
DROP DATABASE IF EXISTS darg_database;

CREATE DATABASE darg_database;
USE darg_database;

CREATE TABLE tb_clientes
```

2. Declaración de restricciones y FK fuera de la tabla

- Las restricciones de cada campo, así como las claves foráneas (FK), se declaran fuera de la tabla. Esta práctica permite una mejor organización del código y mejora la claridad de la estructura de la base de datos.

```sql
/**UNIQUE**/
ALTER TABLE tb_clientes
ADD CONSTRAINT u_dui_cliente UNIQUE (dui_cliente),
ADD CONSTRAINT u_telefono_cliente UNIQUE (telefono_cliente),
ADD CONSTRAINT u_correo_cliente UNIQUE (correo_cliente),
ADD CONSTRAINT u_NIT_cliente UNIQUE (NIT_cliente),
ADD CONSTRAINT u_NRC_cliente UNIQUE (NRC_cliente),
ADD CONSTRAINT u_NRF_cliente UNIQUE (NRF_cliente);
```

```sql
ADD CONSTRAINT u_fk_cliente_automovil FOREIGN KEY (id_cliente) REFERENCES tb_clientes(id_cliente);
```

d. Comentarios en el código

- El uso de comentarios es para proporcionar claridad y documentación dentro del código. Se utiliza en todos los archivos para explicar el propósito de funciones, variables y secciones del código, así como para anotar consideraciones importantes. Se siguen las convenciones de comentarios según el lenguaje, por ejemplo, el uso de // para comentarios de una sola línea y /* ... */ para comentarios de varias líneas.

10. Requerimientos de hardware y software.

a. Hardware de la Computadora:

- **Procesador**: Intel Core i3 o equivalente (o superior).
- **Memoria RAM**: Mínimo 4 GB (preferible 8 GB o más).
- **Almacenamiento**: Mínimo 100GB de espacio libre en disco (HDD o SSD).
- **Tarjeta Gráfica**: Integrada o dedicada (no es crítico, pero puede mejorar la experiencia visual).
- **Conectividad**: Conexión a Internet (Ethernet o Wi-Fi).

b. Software:

- **Sistema Operativo**: Windows 10 o superior, macOS Mojave o superior, o cualquier distribución de Linux moderna.
- **Navegador Web**:
    - Google Chrome (última versión).
    - Mozilla Firefox (última versión).
    - Microsoft Edge (última versión).
    - Safari (solo en macOS).

c. 1. XAMPP

XAMPP proporciona un entorno de servidor web con Apache, lo que permite ejecutar la aplicación en una computadora local durante el desarrollo y pruebas. Con **Apache**, podemos servir las páginas web y gestionar las solicitudes HTTP, como las que procesan la creación de usuarios, el manejo de citas, y la actualización del estado de los automóviles.

d. 2. MariaDB

Para manejar la base de datos, utilizamos **MariaDB**, que se incluye en **XAMPP**. **MariaDB** gestiona todos los datos del sistema, como la información de los clientes, los detalles de los automóviles, y las citas programadas. Las tablas en **MariaDB** almacenan estos datos y se integran con el sistema a través

de consultas SQL en los scripts PHP. Esto permite, por ejemplo, que un administrador cree nuevos usuarios y asigne carros a esos usuarios registrados.

11. Instalación y configuración.

- Descarga e instalación de Visual Studio Code:
  - Visita el sitio oficial de Visual Studio Code.
  - Haz clic en el botón de descarga correspondiente a tu sistema operativo (Windows, macOS o Linux).
  - Ejecuta el archivo descargado.
  - Acepta el acuerdo de licencia y haz clic en "Siguiente".
  - Elige la ubicación de instalación o deja la predeterminada y haz clic en "Siguiente".
  - Selecciona las opciones adicionales que desees, como crear accesos directos, y haz clic en "Siguiente".
  - Haz clic en "Instalar" y espera a que finalice la instalación.
  - Una vez completada, puedes seleccionar "Iniciar Visual Studio Code" y hacer clic en "Finalizar".
  - Puedes instalar extensiones según sea necesario para tu desarrollo, como soporte para PHP, HTML, CSS, etc. Para ello, ve a la pestaña de extensiones (icono de cuadrado en la barra lateral) y busca las extensiones deseadas.

- Instalación de PHP:
  - PHP se incluye en XAMPP, aunque también está disponible en su sitio oficial (PHP, 2024).
- Bootstrap
  - Versión recomendada: Bootstrap 5.x
  - Incorporación en HTML:
  - <script src=”../../api/libs/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
- Archivos locales:
  - Asegúrate de incluir los archivos bootstrap.bundle.min.js y el CSS correspondiente en tu proyecto.
- XAMPP
  - Visita el sitio oficial de XAMPP.

- En la página de inicio, encontrarás diferentes versiones de XAMPP. Haz clic en el botón de descarga correspondiente a tu sistema operativo (Windows, macOS o Linux).
- Ejecuta el archivo descargado.
- En la ventana del instalador, selecciona los componentes que deseas instalar (Apache, MySQL, PHP, etc.). Puedes dejar las opciones predeterminadas seleccionadas.
- Haz clic en "Next".
- Elige la carpeta de instalación o deja la predeterminada (generalmente C:\xampp) y haz clic en "Next".
- Puedes desmarcar la opción de iniciar el panel de control de XAMPP al final de la instalación si no deseas abrirlo de inmediato.
- Haz clic en "Next" y luego en "Finish" para completar la instalación.
- Verifica la correcta configuración de Apache y MySQL.

- SweetAlert2
  - Versión recomendada: SweetAlert 11.x
  - Incorporación en HTML:
  - <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
- Chart.js
  - Versión recomendada: Chart.js 3.x o superior.
  - Incorporación en HTML:
  - <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
- FPDF
  - Versión: FPDF 1.85
  - Instalación:
  - Descárgalo desde el sitio oficial de FPDF.
  - Incorporación:
  - Incluye el archivo fpdf.php en tu proyecto.
- Conexión de Bibliotecas:
  - Para integrar Bootstrap, SweetAlert y Chart.js, asegúrate de incluir las referencias adecuadas utilizando las etiquetas <script> y verificando que las rutas en tu HTML sean correctas:
  - <script src="../../api/libs/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

- <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
- <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

a. Configuración en servidor web:

- Paso 1: Primero, debes abrir una cuenta en Google Cloud, donde se alojará el servidor utilizando una máquina virtual con una distribución Debian Linux. Durante los primeros tres meses, el servicio es gratuito, pero después se realizará un cargo automático en la tarjeta registrada en el proceso de creación de cuenta.
- Paso 2: Una vez en Google Cloud, crea una nueva instancia de máquina virtual con el sistema operativo Debian GNU/Linux 11 (Bullseye). Asegúrate de habilitar el tráfico HTTP y HTTPS para permitir el acceso web a la instancia.
- Paso 3: Conéctate a la máquina virtual utilizando SSH, que te proporcionará acceso a la terminal. Desde allí, instala MySQL utilizando los comandos correspondientes y configura un usuario con los permisos adecuados para manejar la base de datos.
- Paso 4: Instala el servidor web Apache de la misma manera que instalaste MySQL, utilizando comandos en la terminal. Este servidor será el encargado de procesar las peticiones web.
- Paso 5: Sube los archivos del proyecto a la máquina virtual en un archivo comprimido (ZIP). Instala las herramientas necesarias para manipular archivos comprimidos y descomprime los archivos en el directorio /var/www/html, que es la ubicación predeterminada para los sitios web en Apache.
- Paso 6: Ingresa a MySQL ejecutando el comando mysql -u root -p. Desde allí, crea la base de datos, las tablas necesarias, triggers, vistas, y cualquier otra estructura que tu proyecto requiera.
- Paso 7: Abre y edita los archivos config.php y components.js de tu proyecto, que se encuentran en la carpeta que acabas de descomprimir. En config.php, reemplaza las credenciales de conexión y la dirección de localhost por la IP del servidor. Asegúrate de hacer lo mismo en cualquier otro archivo de configuración relacionado.
- Paso 8: Enlaza un dominio a tu servidor en Google Cloud para que el proyecto esté accesible a través de un nombre de dominio en lugar de solo la dirección IP.
- Paso 9: Finalmente, instala un certificado SSL en el servidor para asegurar el tráfico HTTPS, lo cual mejorará la seguridad de la comunicación entre los usuarios y el servidor.

b. Localmente

- Instalar un servidor web local: Para ejecutar un sistema web en tu máquina local, necesitarás instalar un entorno de servidor local. Las opciones más comunes son:
- Opción 1: XAMPP (Windows, macOS, Linux): XAMPP incluye Apache (servidor web), MySQL (gestor de bases de datos), y PHP (lenguaje de servidor). Es una opción fácil para comenzar con un entorno de desarrollo web.
- Pasos:
- Descarga XAMPP desde el sitio oficial: https://www.apachefriends.org/index.html.
- Instala XAMPP siguiendo las instrucciones.
- Abre el panel de control de XAMPP y arranca los servicios de Apache (servidor web) y MySQL (base de datos).
- Luego para usar el sistema, en tu buscador busca http://localhost/proyecto-tecnico-cientifico-2024/vistas/privado/pagina_servicios.html y ya se podría usar