INSTITUTO TÉCNICO RICALDONE

TERCER AÑO DE BACHILLERATO

DESARROLLO DE SOFTWARE

Manual técnico para el proyecto técnico científico

Asignatura:

Elaboración de documentación de sistemas informáticos

Desarrolladores:

Guillermo Stanley, Recinos Alvarenga	20220383
Diego Fernando, Ortiz Flores	20190629
Paola Beatriz, Martínez Sánchez	20220490
Rafael José, Paniagua González	20220100

Docente:

Antonio Samuel Benavides Rivas

Índice

Introducción

El presente manual técnico tiene como objetivo proporcionar una guía detallada sobre el desarrollo y la implementación del sistema web. Este documento está dirigido a los desarrolladores, administradores y usuarios del sistema, así como a cualquier persona interesada en comprender la estructura y funcionamiento del software. En la primera sección, se describen las tecnologías y herramientas empleadas, donde se presentan los diferentes componentes técnicos que han sido utilizados, incluyendo lenguajes de programación, frameworks y bases de datos. A continuación, se expone la estructura de la base de datos, mostrando el modelo relacional y proporcionando el script SQL necesario para su construcción. El diccionario de datos proporciona una referencia clara sobre los elementos que conforman la base de datos, facilitando así la comprensión de su funcionamiento. Posteriormente, se aborda la arquitectura del software, donde se detalla la interacción entre las distintas capas del sistema. En la sección sobre la estructura del proyecto, se describe la organización de los directorios y archivos, proporcionando un panorama de cómo están dispuestos los componentes del sistema. El diseño de la aplicación es también fundamental, ya que se presentarán las características visuales y estéticas que definen la interfaz de usuario. Además, se incluyen las buenas prácticas de desarrollo, que garantizan la calidad y mantenibilidad del código, así como los requerimientos de hardware y software necesarios para la correcta operación del sistema. Finalmente, se detalla el proceso de instalación y configuración, tanto en entornos locales como en línea, para asegurar que los usuarios puedan poner en marcha el sistema sin inconvenientes.

Tecnologías y herramientas empleadas

- Servidor Web
    - XAMPP
    - Versión: 8.2.0
    - Descripción: Paquete que incluye Apache, PHP y MariaDB para crear un entorno de desarrollo local para aplicaciones web.
    - Sitio web oficial: https://www.apachefriends.org/es/index.html
- Lenguajes de Programación
    - **Del lado del servidor:**
    - PHP
    - Versión: 8.2.0
    - Descripción: Lenguaje de programación utilizado para la lógica del lado del servidor.
    - Sitio web oficial: https://www.php.net
    - **Del lado del cliente:**
    - JavaScript
    - Descripción: Lenguaje de programación utilizado para añadir interactividad a la interfaz del usuario.
    - Sitio web oficial: https://www.javascript.com

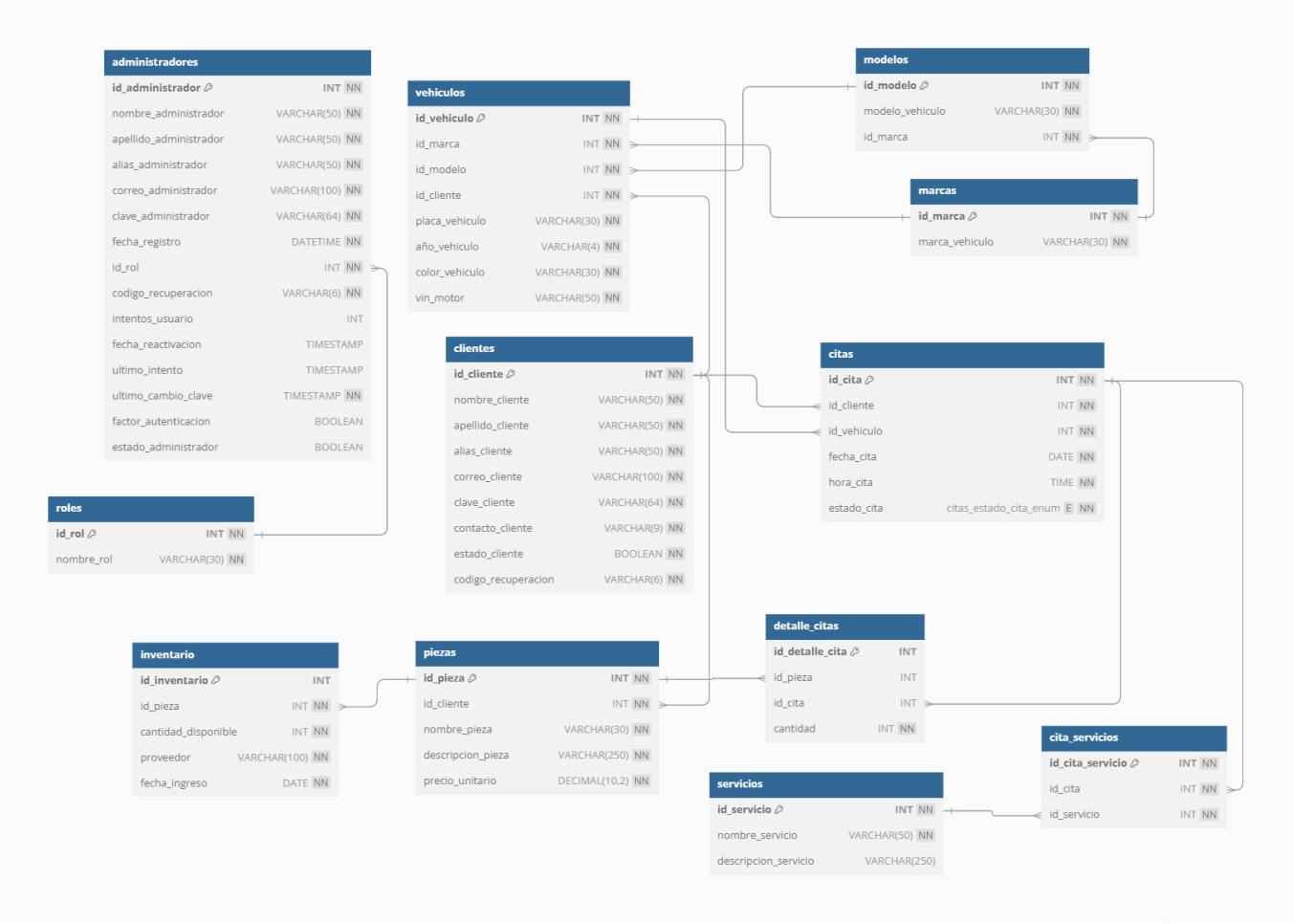
- Base de Datos
    - MariaDB

    - Versión: 10.4.27
    - Descripción: Sistema de gestión de bases de datos relacional utilizado para almacenar la información de la tienda.
    - Sitio web oficial: https://mariadb.org
- Frameworks
    - Bootstrap
    - Versión: 5.3.0
    - Descripción: Framework CSS que facilita el diseño de interfaces responsivas y atractivas.
    - Sitio web oficial: https://getbootstrap.com
- Librerías
    - FPDF
        - Versión: 1.85
        - Descripción: Librería para la generación de archivos PDF desde PHP.
        - Sitio web oficial: http://www.fpdf.org/
    - Bootstrap Icons
        - Versión: 1.10.5
        - Descripción: Conjunto de íconos que complementan Bootstrap, mejorando la interfaz visual.
        - Sitio web oficial: https://icons.getbootstrap.com/
    - SweetAlert
        - Versión: 2.0

        - Descripción: Librería que permite crear alertas personalizadas y atractivas en JavaScript.
        - Sitio web oficial: https://sweetalert.js.org/
    - Chart.js
        - Versión: 4.3.0
        - Descripción: Librería de JavaScript para crear gráficos interactivos y visuales.
        - Sitio web oficial: https://www.chartjs.org/
- IDEs y Editores de Texto
    - Visual Studio Code
        - Descripción: Editor de código fuente que proporciona soporte para múltiples lenguajes de programación y herramientas de desarrollo.
        - Sitio web oficial: https://code.visualstudio.com

Estructura de la base de datos

- Modelo relacional:



- Script SQL:

```
DROP DATABASE IF EXISTS db_taller_rodriguez;


CREATE DATABASE db_taller_rodriguez;
```

```
USE db_taller_rodriguez;


CREATE table roles(
id_rol INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
nombre_rol VARCHAR(30) NOT NULL
);


CREATE TABLE administradores(
    id_administrador INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    nombre_administrador VARCHAR(50) NOT NULL,
    apellido_administrador VARCHAR(50) NOT NULL,
    alias_administrador VARCHAR(50) NOT NULL,
    correo_administrador VARCHAR(100) NOT NULL,
    clave_administrador VARCHAR(64) NOT NULL,
    fecha_registro DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP(),
    id_rol INT NOT NULL,
    CONSTRAINT fk_rol_administrador
    FOREIGN KEY (id_rol)
    REFERENCES roles (id_rol) ON DELETE CASCADE,
```

```
    codigo_recuperacion VARCHAR(6) NOT NULL,
    intentos_usuario INT UNSIGNED DEFAULT 0,
    fecha_reactivacion TIMESTAMP NULL DEFAULT NULL,
    ultimo_intento TIMESTAMP NULL DEFAULT NULL,
    ultimo_cambio_clave TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    factor_autenticacion BOOLEAN DEFAULT FALSE,
    estado_administrador BOOLEAN DEFAULT TRUE
);

ALTER TABLE administradores
ADD CONSTRAINT unique_correo_administrador UNIQUE (correo_administrador);

ALTER TABLE administradores
ADD CONSTRAINT unique_alias_administrador UNIQUE (alias_administrador);

CREATE TABLE marcas(
    id_marca INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    marca_vehiculo VARCHAR(30) NOT NULL
);
```

```
ALTER TABLE marcas

ADD CONSTRAINT unique_marca_vehiculo UNIQUE (marca_vehiculo);


CREATE TABLE modelos(

   id_modelo INT PRIMARY KEY AUTO_INCREMENT NOT NULL,

   modelo_vehiculo VARCHAR(30) NOT NULL,

   id_marca INT NOT NULL,

   CONSTRAINT fk_modelo_marca

   FOREIGN KEY (id_marca)

   REFERENCES marcas (id_marca) ON DELETE CASCADE

);


ALTER TABLE modelos

ADD CONSTRAINT unique_modelo_vehiculo UNIQUE (modelo_vehiculo, id_marca);


CREATE TABLE clientes(

   id_cliente INT PRIMARY KEY AUTO_INCREMENT NOT NULL,

   nombre_cliente VARCHAR(50) NOT NULL,
```

```
    apellido_cliente VARCHAR(50) NOT NULL,
    alias_cliente VARCHAR(50) NOT NULL,
    correo_cliente VARCHAR(100) NOT NULL,
    clave_cliente VARCHAR(64) NOT NULL,
    contacto_cliente VARCHAR(9) NOT NULL,
    estado_cliente BOOLEAN NOT NULL,
    codigo_recuperacion VARCHAR(6) NOT NULL
);

ALTER TABLE clientes
ADD CONSTRAINT chk_contacto_cliente_format
CHECK (contacto_cliente REGEXP '^[0-9]{4}-[0-9]{4}$');

ALTER TABLE clientes
ADD CONSTRAINT unique_correo_cliente UNIQUE (correo_cliente);

ALTER TABLE clientes
ADD CONSTRAINT unique_alias_cliente UNIQUE (alias_cliente);
```

```
ALTER TABLE clientes

ALTER COLUMN estado_cliente SET DEFAULT 1;



CREATE TABLE vehiculos(

    id_vehiculo INT PRIMARY KEY AUTO_INCREMENT NOT NULL,

    id_marca INT NOT NULL,

    id_modelo INT NOT NULL,

    id_cliente INT NOT NULL,

    placa_vehiculo VARCHAR(30) NOT NULL,

    año_vehiculo VARCHAR(4) NOT NULL,

    color_vehiculo VARCHAR(30) NOT NULL,

    vin_motor VARCHAR(50) NOT NULL,

    CONSTRAINT fk_vehiculo_marca

    FOREIGN KEY (id_marca)

    REFERENCES marcas (id_marca) ON DELETE CASCADE,

    CONSTRAINT fk_vehiculo_modelo

    FOREIGN KEY (id_modelo)

    REFERENCES modelos (id_modelo) ON DELETE CASCADE,

    CONSTRAINT fk_vehiculo_cliente
```

```
    FOREIGN KEY (id_cliente)
    REFERENCES clientes (id_cliente) ON DELETE CASCADE
);

CREATE TABLE servicios(
    id_servicio INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    nombre_servicio VARCHAR(50) NOT NULL,
    descripcion_servicio VARCHAR(250)
);

CREATE TABLE citas(
    id_cita INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    id_cliente INT NOT NULL,
    id_vehiculo INT NOT NULL,
    fecha_cita DATE NOT NULL,
    hora_cita TIME NOT NULL,
    estado_cita ENUM('Pendiente', 'Completada', 'Cancelada') NOT NULL,
    CONSTRAINT fk_cita_cliente
    FOREIGN KEY (id_cliente)
```

```
    REFERENCES clientes (id_cliente) ON DELETE CASCADE,
    CONSTRAINT fk_cita_vehiculo
    FOREIGN KEY (id_vehiculo)
    REFERENCES vehiculos (id_vehiculo) ON DELETE CASCADE
);

ALTER TABLE citas
ALTER COLUMN estado_cita SET DEFAULT 'Pendiente';

CREATE TABLE cita_servicios(
    id_cita_servicio INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    id_cita INT NOT NULL,
    id_servicio INT NOT NULL,
    CONSTRAINT fk_cita_servicio_cita
    FOREIGN KEY (id_cita)
    REFERENCES citas (id_cita) ON DELETE CASCADE,
    CONSTRAINT fk_cita_servicio_servicio
    FOREIGN KEY (id_servicio)
    REFERENCES servicios (id_servicio) ON DELETE CASCADE
```

```
);

DESCRIBE cita_servicios;

CREATE TABLE piezas(
    id_pieza INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    id_cliente INT NOT NULL,
    nombre_pieza VARCHAR(30) NOT NULL,
    descripcion_pieza VARCHAR(250) NOT NULL,
    precio_unitario DECIMAL(10, 2) NOT NULL,
    CONSTRAINT fk_pieza_vehiculo
    FOREIGN KEY (id_cliente)
    REFERENCES clientes (id_cliente) ON DELETE CASCADE
);

ALTER TABLE piezas
ADD CONSTRAINT unique_nombre_pieza UNIQUE (nombre_pieza);

ALTER TABLE piezas
```

```
ADD CONSTRAINT chk_precio_unitario_positive CHECK (precio_unitario > 0);


CREATE TABLE inventario (
    id_inventario INT AUTO_INCREMENT PRIMARY KEY,
    id_pieza INT NOT NULL,
    cantidad_disponible INT NOT NULL,
    proveedor VARCHAR(100) NOT NULL,
    fecha_ingreso DATE NOT NULL,
    CONSTRAINT fk_inventario_pieza
    FOREIGN KEY (id_pieza)
    REFERENCES piezas (id_pieza) ON DELETE CASCADE
);


CREATE TABLE detalle_citas(
id_detalle_cita INT AUTO_INCREMENT PRIMARY KEY,
id_pieza INT,
 CONSTRAINT fk_detalle_pieza
    FOREIGN KEY (id_pieza)
    REFERENCES piezas (id_pieza) ON DELETE CASCADE,
```

```
    id_cita INT,
    CONSTRAINT fk_detalle_cita
    FOREIGN KEY (id_cita)
    REFERENCES citas (id_cita) ON DELETE CASCADE,
    cantidad INT NOT NULL
);
```

Diccionario de datos

- Tabla: roles

COLUMNA	TIPO	LONGITUD	NULO	ÍNDICE	PREDETERMINADO
id_rol	INT	10	No	Llave Primaria	Autonumérico
nombre_rol	VARCHAR	30	No		

- Tabla: administradores

COLUMNA	TIPO	LONGITUD	NULO	ÍNDICE	PREDETERMINADO
id_administrador	INT	10	No	Llave Primaria	Autonumérico
nombre_administrador	VARCHAR	50	No		
apellido_administrador	VARCHAR	50	No		
alias_administrador	VARCHAR	50	No	Único	
correo_administrador	VARCHAR	100	No	Único	
clave_administrador	VARCHAR	64	No		
fecha_registro	DATETIME		No		CURRENT_TIMESTAMP
id_rol	INT	10	No	Llave foránea	
codigo_recuperacion	VARCHAR	6	No		
intentos_usuario	INT UNSIGNED		Sí		

fecha_reactivacion	TIMESTAMP		Sí		NULL
ultimo_intento	TIMESTAMP		Sí		NULL
ultimo_cambio_clave	TIMESTAMP		No		CURRENT_TIMESTAMP
factor_autenticacion	BOOLEAN		Sí		FALSE
estado_administrador	BOOLEAN		No		TRUE

- Tabla: marcas

COLUMNA	TIPO	LONGITUD	NULO	ÍNDICE	PREDETERMINADO
id_marca	INT	10	No	Llave Primaria	Autonumérico
marca_vehiculo	VARCHAR	30	No	Único	

- Tabla: modelos

COLUMNA	TIPO	LONGITUD	NULO	ÍNDICE	PREDETERMINADO
id_modelo	INT	10	No	Llave Primaria	Autonumérico
modelo_vehiculo	VARCHAR	30	No	Único	
id_marca	INT	10	No	Llave foránea	

- Tabla: clientes

COLUMNA	TIPO	LONGITUD	NULO	ÍNDICE	PREDETERMINADO
id_cliente	INT	10	No	Llave Primaria	Autonumérico
nombre_cliente	VARCHAR	50	No		
apellido_cliente	VARCHAR	50	No		
alias_cliente	VARCHAR	50	No	Único	
correo_cliente	VARCHAR	100	No	Único	
clave_cliente	VARCHAR	64	No		
contacto_cliente	VARCHAR	9	No		
estado_cliente	BOOLEAN		No		
codigo_recuperacion	VARCHAR	6	No		

- Tabla: vehiculos

COLUMNA	TIPO	LONGITUD	NULO	ÍNDICE	PREDETERMINADO
id_vehiculo	INT	10	No	Llave Primaria	Autonumérico
id_marca	INT	10	No	Llave foránea	
id_modelo	INT	10	No	Llave foránea	

id_cliente	INT	10	No	Llave foránea	
placa_vehiculo	VARCHAR	30	No		
año_vehiculo	VARCHAR	4	No		
color_vehiculo	VARCHAR	30	No		
vin_motor	VARCHAR	50	No		

- Tabla: servicios

COLUMNA	TIPO	LONGITUD	NULO	ÍNDICE	PREDETERMINADO
id_servicio	INT	10	No	Llave Primaria	Autonumérico
nombre_servicio	VARCHAR	50	No		
descripcion_servicio	VARCHAR	250	Sí		

- Tabla: citas

COLUMNA	TIPO	LONGITUD	NULO	ÍNDICE	PREDETERMINADO
id_cita	INT	10	No	Llave Primaria	Autonumérico
id_cliente	INT	10	No	Llave foránea	
id_vehiculo	INT	10	No	Llave foránea	
fecha_cita	INT		No		
hora_cita	TIME		No		
estado_cita	ENUM		No		'Pendiente'

- Tabla: cita_servicios

COLUMNA	TIPO	LONGITUD	NULO	ÍNDICE	PREDETERMINADO
id_cita_servicio	INT	10	No	Llave Primaria	Autonumérico
id_cita	INT	10	No	Llave foránea	
id_servicio	INT	10	No	Llave foránea	

- Tabla: piezas

COLUMNA	TIPO	LONGITUD	NULO	ÍNDICE	PREDETERMINADO
id_pieza	INT	10	No	Llave Primaria	Autonumérico
id_cliente	INT	10	No	Llave foránea	
nombre_pieza	VARCHAR	30	No	Único	
descripcion_pieza	VARCHAR	250	No		
precio_unitario	DECIMAL	10,2	No		

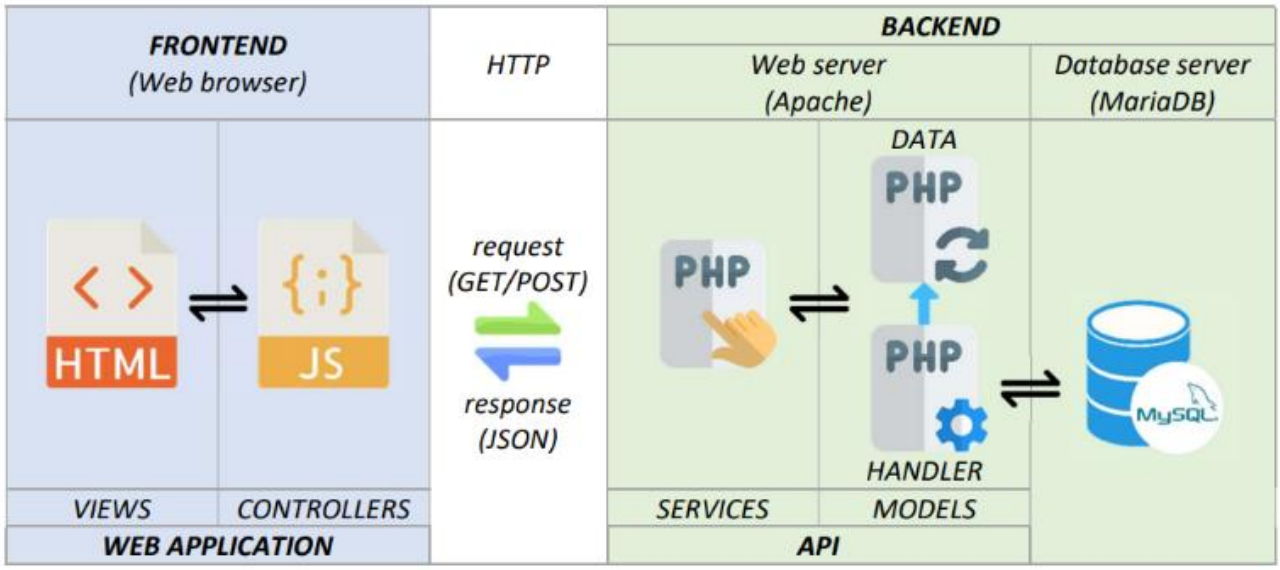
- Tabla: inventario

COLUMNA	TIPO	LONGITUD	NULO	ÍNDICE	PREDETERMINADO
id_inventario	INT	10	No	Llave Primaria	Autonumérico
id_pieza	INT	10	No	Llave foránea	
cantidad_disponible	INT		No		
proveedor	VARCHAR	100	No		
fecha_ingreso	DATE		No		

- Tabla: detalle_citas

COLUMNA	TIPO	LONGITUD	NULO	ÍNDICE	PREDETERMINADO
id_detalle_cita	INT	10	No	Llave Primaria	Autonumérico
id_pieza	INT	10	Sí	Llave foránea	
id_cita	INT	10	Sí	Llave foránea	
cantidad	INT		No		

Arquitectura del software



- Vistas (Views)

Las vistas (views) son prácticamente páginas web elaboradas con HTML5 empleando Bootstrap, las cuales interactúan directamente con los controladores (controllers) programados en JavaScript mediante Vanilla JS. En términos generales, la programación asíncrona con JavaScript permite realizar peticiones y recibir respuestas en tiempo real para manipular el DOM (Document Object Model) según las necesidades de la aplicación, es decir, es lo que hace funcionar el contenido dinámico en las páginas web. La programación asíncrona facilita el manejo de la interacción del usuario con las vistas sin recargar la página web, o sea, sin refrescar el navegador.

- Peticiones y Respuestas

Por medio de los controladores se hacen peticiones (request) y se reciben respuestas (response) en formato JSON (JavaScript Object Notation). Cuando una petición implica enviar datos, como por ejemplo actualizar un registro, se remiten por medio del método POST; en cambio, si lo que se requiere únicamente es solicitar datos, como por ejemplo listar registros, se piden a través del método GET.

- Funcionamiento del Frontend

Las vistas y los controladores son los únicos elementos que funcionan del lado del cliente (frontend) interpretados por el navegador web, donde la implementación de programación asíncrona puede realizarse de diversas formas, no solo con Vanilla JS. Por ejemplo, se podría optar por utilizar librerías como jQuery (no recomendable), React, entre otras; o dar un salto más allá mediante la utilización de un framework basado en JavaScript como Vue, Angular u otro similar.

- Controladores (Controllers)

Para cada página web existe un archivo JavaScript que actúa como controlador principal, capaz de gestionar todas las peticiones y respuestas, ya sea de forma automática o por intervención del usuario. Dicha gestión se realiza a través de diversos métodos, eventos y funciones; que dotan a la página web de toda la funcionalidad necesaria para realizar una o varias tareas específicas. Por conveniencia, se recomienda que el nombre del controlador principal sea el de la página web.

- Backend y API

Por otro lado, tenemos los elementos del lado del servidor (backend) que llamaremos API (Application Programming Interface – Interfaz de Programación de Aplicaciones), la cual interactúa directamente con los controladores por medio de la capa de servicios (SERVICES), recibiendo peticiones GET o POST y retornando respuestas en formato JSON. La API reacciona según las acciones requeridas por los controladores, donde cada acción está definida en la URL del archivo que la contiene, lo que recibe el nombre de endpoint.

- Programacion de la API

La API está programada con PHP, instanciando en SERVICES una o más clases de los modelos (MODELS), es decir, se implementa OOP (Object Oriented Programming – Programación Orientada a Objetos). Además, en los modelos se validan todos los datos de entrada del lado del servidor al momento de establecer (SET) los valores a los atributos, específicamente en la capa DATA, como parte del encapsulamiento por si se escapa algún dato erróneo o malicioso desde el frontend.

- Capa de Servicios

La capa SERVICES está formada por varios archivos o scripts, cada archivo contiene una o varias acciones (endpoints). Algunas acciones pueden requerir autenticación del usuario, por lo que se debe validar para evitar exponer datos sensibles o realizar operaciones sin control, previniendo de esta forma posibles ataques o robos de información. También, aquí se obtienen las excepciones de la base de datos y se realiza el manejo de errores en los valores de entrada.

- Modelos (Models)

Los modelos son una representación fidedigna de la base de datos, modelando generalmente cada tabla por medio de dos clases: DATA y HANDLER. Cada clase debe elaborarse en un archivo, o por lo menos eso sugieren las PSR (PHP Standards Recommendations – Recomendaciones de Estándares PHP). En dichos modelos se realizan las operaciones con la base de datos por medio de la capa HANDLER, donde se implementa la clase PDO (PHP Data Object – Objetos de Datos PHP).

- Uso de PDO

La extensión PDO es una clase nativa en PHP a partir de la versión 5.1, la cual define una interfaz ligera para trabajar con diferentes bases de datos a través de controladores específicos que proveen características propias de cada base de datos, siendo MariaDB una de las opciones soportadas. PDO proporciona una capa de abstracción de acceso a datos, lo que significa que puede utilizarse con diversos gestores de bases de datos de forma automática o realizando pequeñas configuraciones.

- Proceso de Desarrollo

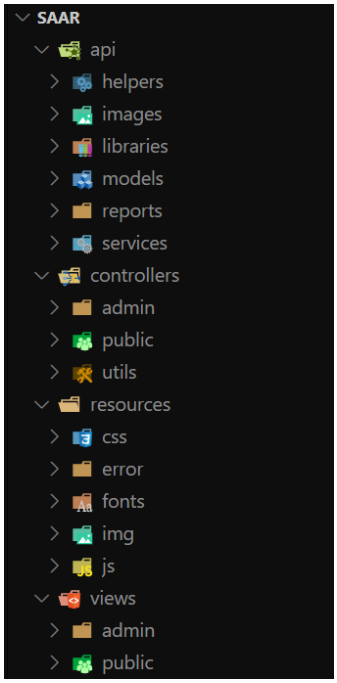
Ahora viene la pregunta: ¿por dónde se empieza? No existe una respuesta única, sin embargo, se puede seguir cierta lógica para llevar a cabo el desarrollo, tal como se sugiere a continuación:

  - Construir la base de datos.
  - Elaborar los mockups definiendo todos los detalles de diseño y tomando en cuenta la base de datos.

    - Crear las vistas (views) con un framework de acuerdo con los mockups.
    - Programar las clases de los modelos (models) con PHP según la base de datos:
        - Clase Handler: definir las propiedades a partir de las columnas de la tabla y luego un método por cada sentencia SQL.
        - Clase Data: crear un método set por cada propiedad e incluir la validación respectiva antes de asignar el valor.
- Implementar los modelos para las acciones de los servicios (services) y probar con Postman o una herramienta similar.
- Programar los controladores (controllers) con JavaScript.

Estructura del proyecto

- **api:** esta carpeta es la interfaz de programación de la aplicación del lado del servidor, donde services contiene toda la lógica del negocio para recibir peticiones (request) y enviar resultados (response), models cuenta con todo lo relacionado al manejo de los datos, helpers sirve para albergar funcionalidad de uso general como por ejemplo: validaciones, conexión a la base de datos, plantillas, etc.; images guarda todos los archivos de imágenes relacionadas con los datos, libraries aloja librerías de terceros como por ejemplo: PHPMailer y fpdf, las cuales sirven para ampliar la funcionalidad de la aplicación; y reports contiene toda la programación necesaria para generar reportes a partir de la base de datos.
- **controllers:** representa la lógica de programación del lado del cliente que interactúa con la API, es decir, realiza peticiones y recibe resultados. Las carpetas admin y public cuentan con los controladores necesarios para cada sitio web y utils es para alojar scripts de uso general.
- **resources:** este directorio contiene esencialmente todos aquellos archivos estáticos de propósito general, ya sean propios o de terceros, es decir: hojas de estilo en cascada (css), fuentes tipográficas (fonts), imágenes (img), scripts (js) y páginas web personalizadas para el manejo de errores (error).
- **views:** aquí es donde se encuentran todas las páginas web de la aplicación, es decir, las vistas. Dentro de este directorio se ha creado una carpeta para el sitio público (public) y otra para el sitio privado (admin).

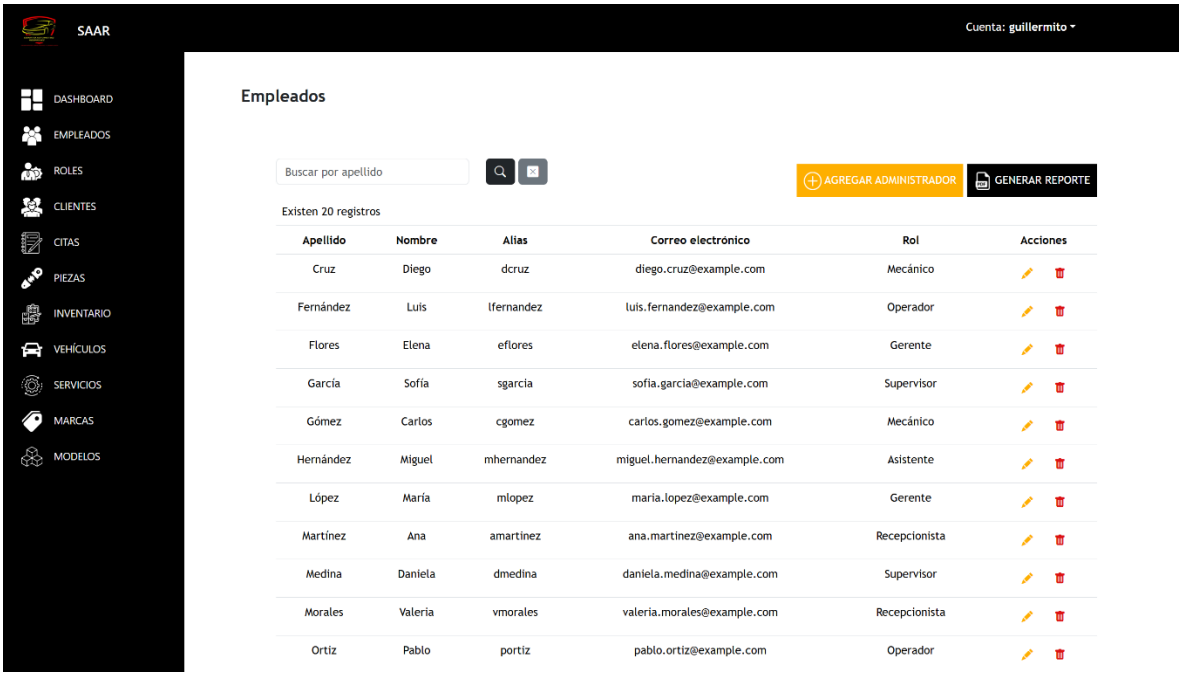

SAAR
api
helpers
images
libraries
models
reports
services
controllers
admin
public
utils
resources
css
error
fonts
img
js
views
admin
public

Diseño de la aplicación

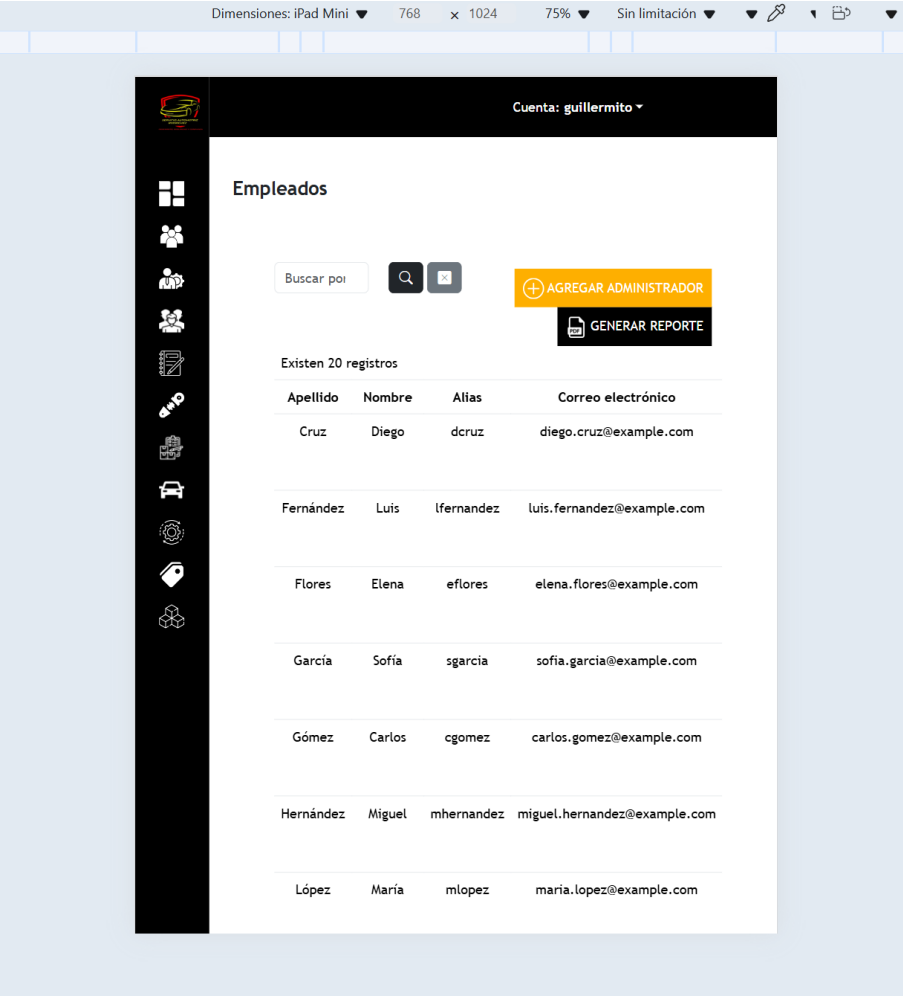
- Paleta de colores:

Paleta de colores

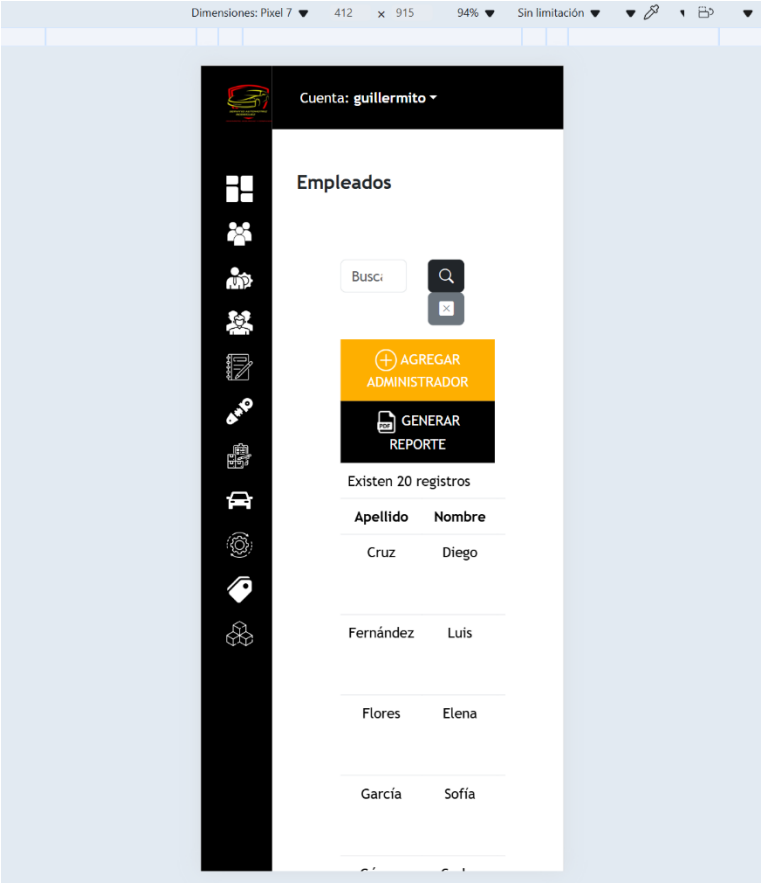
Elemento de referencia:



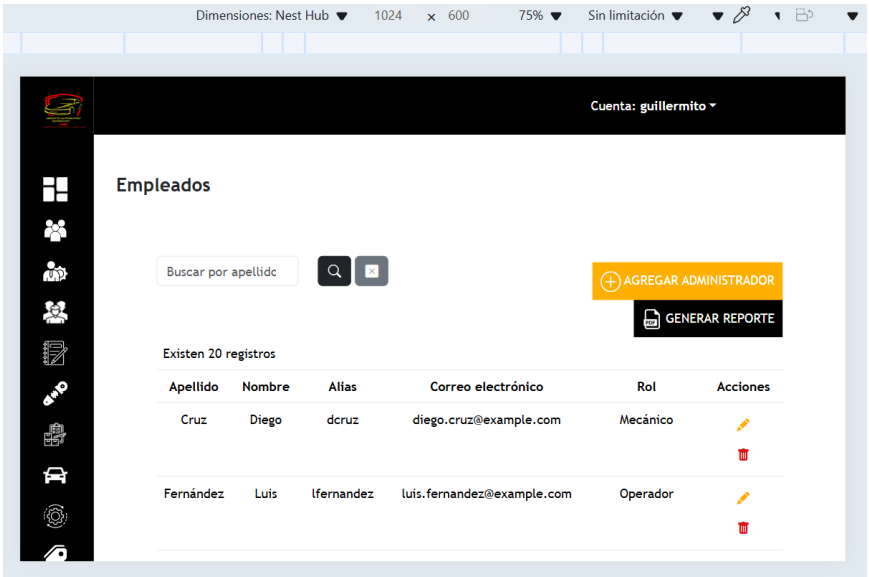
- Fuentes:
    - 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans'
- Tamaños de fuentes:
    - Titulos: 24px
    - Texto: 16px
- Tamaños de pantalla:
    - iPad Mini:



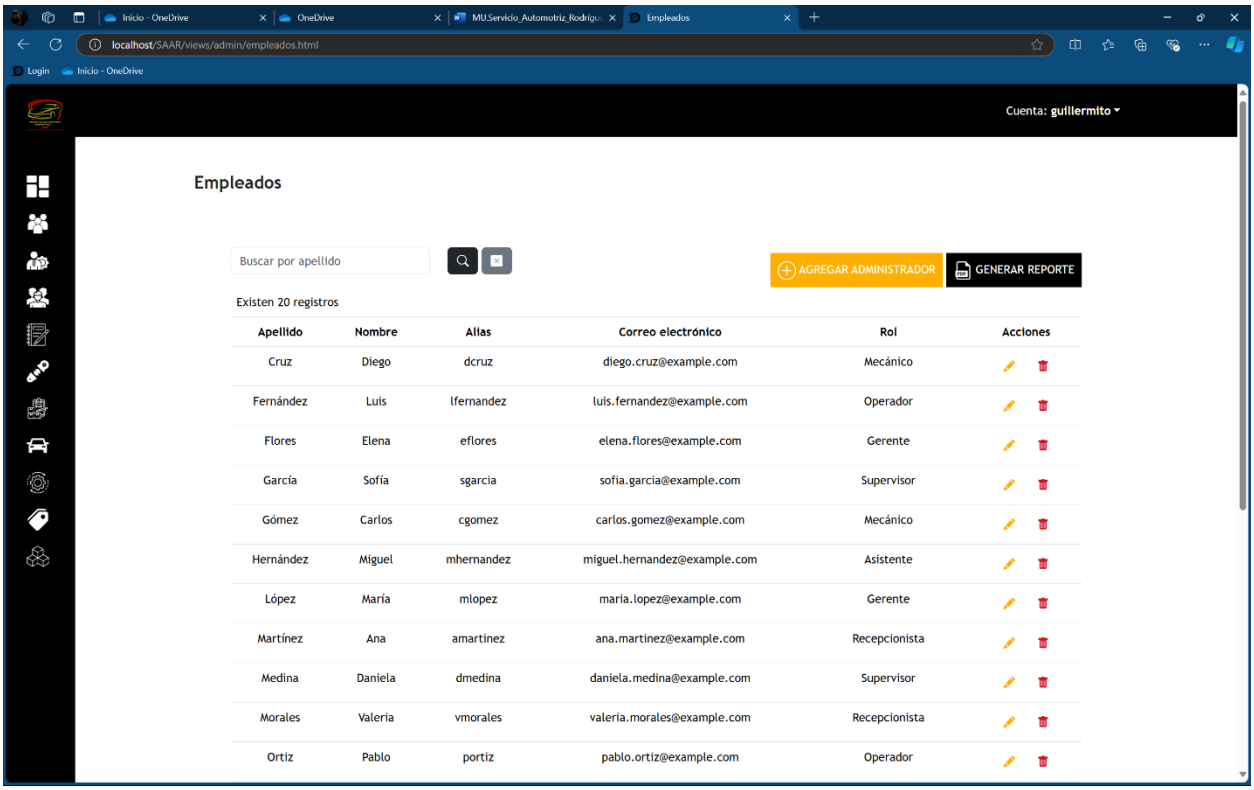
- Pixel 7:



- Nest Hub:



- IdeaPad Flex 5 161AU7:



Buenas prácticas de desarrollo

- PHP:
    - Los archivos php deben iniciar con la etiqueta "<?php"
    - El php debe estar escrito en el formato de codificacion UTF-8
    - Un archivo php debe cuidar si usará el archivo para generar conexiones con "require" o para determinar nuevas clases o funciones,
    - pero nunca deberá usar los 2 en el mismo archivo, el condicional no tiene efectos secundarios
    - Las clases deben estar escritas en StudlyCaps
    - Los métodos tienen que estar escritos en camelCase
- JavaScript:
    - Los archivos tienen que estar codificados en UTF-8
    - Los nombres de las constantes deben estar escritas en mayúscula y separado con guiones bajos
    - Los nombres de las funciones deben estar escritas en camelCase
    - No usar Tab para sangría del código, mejor usar la tecla "espacio"
    - Terminar cada párrafo de código con un punto y coma ";"
    - Al momento de usar corchetes, dejar el corchete de apertura en la primera linea y el corchete de cierre en una línea nueva, abajo del código
    - Evitar líneas de más de 80 palabras

- Estándares para Bases de Datos:

- **Nombres de Tablas y Columnas:**

- Los nombres de tablas deben ser descriptivos y reflejar su propósito.

- Mantener una nomenclatura consistente, optando por singular o plural.

- Evitar abreviaciones confusas; usar nombres completos.

- **Tipos de Datos:**

- Utilizar tipos de datos específicos que se ajusten al valor que contendrán (e.g., `VARCHAR` para texto, `INT` para números).

- Definir límites de tamaño adecuados para columnas.

- **Claves Primarias y Foráneas:**

- Usar `AUTO_INCREMENT` para claves primarias.

- Definir claves foráneas para mantener la integridad referencial.

- **Restricciones:**

- Aplicar `UNIQUE` para asegurar que no haya duplicados en columnas únicas.

- Utilizar `CHECK` para validar valores (formato del contacto).

- **Normalización:**

- Seguir la Tercera Forma Normal (3NF) para eliminar redundancias.

- Dividir datos en múltiples tablas según su relación.

Requerimientos de hardware y software

→ **Requerimientos de Hardware**

- **Condiciones Mínimas:**
  - Procesador: Intel Core i3 o equivalente
  - RAM: 4 GB
  - Espacio en Disco: 500 MB disponibles
  - Conexión a Internet: 1 Mbps (para funcionamiento en línea)
- **Condiciones Óptimas:**
  - Procesador: Intel Core i5 o equivalente
  - RAM: 8 GB o más
  - Espacio en Disco: 1 GB disponibles
  - Conexión a Internet: 10 Mbps o más

→ **Requerimientos de Software**

- **Condiciones Mínimas:**
  - Sistema Operativo: Microsoft Windows 10 o superior
  - XAMPP v8.2.0 (PHP v8.2.0, Apache v2.4.54 y MariaDB v10.4.27)
  - Navegador Web: Google Chrome, Mozilla Firefox o equivalente
- **Condiciones Óptimas:**
  - Sistema Operativo: Microsoft Windows 10 o superior
  - XAMPP v8.2.0 (PHP v8.2.0, Apache v2.4.54 y MariaDB v10.4.27)
  - Navegador Web: Google Chrome o Mozilla Firefox en su última versión
  - Actualizaciones de seguridad aplicadas al sistema operativo y a las aplicaciones

Instalación y configuración

- **Instalación y Configuración en Entorno Local**
    - **Requisitos Previos:**
        - Tener instalado XAMPP v8.2.0 en tu computadora.
- **Pasos:**

1. Descargar el Proyecto:
2. Descarga el archivo del proyecto SAAR desde la fuente proporcionada.
3. Ubicación del Proyecto:
4. Descomprime el archivo descargado y coloca la carpeta del proyecto en el directorio C:/xampp/htdocs/.

- **Iniciar XAMPP:**

1. Abre el Panel de Control de XAMPP.
2. Inicia los módulos de Apache y MySQL.

- **Importar la Base de Datos:**

1. Abre tu navegador y ve a http://localhost/phpmyadmin/.
2. Crea una nueva base de datos llamada db_taller_rodriguez.
3. Haz clic en la base de datos recién creada y selecciona la opción "Importar".
4. Selecciona el archivo db_taller_rodriguez.sql que se encuentra en la carpeta del proyecto y haz clic en "Continuar".

- **Configurar Conexión a la Base de Datos:**

1. Navega a la carpeta del proyecto y abre el archivo api/helpers/config.php.
2. Modifica las credenciales de conexión si es necesario (usuario y contraseña predeterminados para XAMPP son root y una contraseña vacía).

- **Acceder al Proyecto:**

1. Para acceder al sitio privado, abre el navegador y ve a http://localhost/SAAR/views/admin/.

- **Crear un Usuario:**

1. Al ingresar al sitio privado por primera vez, se te pedirá que crees un usuario administrador.

- **Instalación y Configuración en Entorno en Línea**
    - **Requisitos Previos:**
        - Elegir un servicio de alojamiento web gratuito como 000webhost, InfinityFree o Heroku.

- **Pasos:**

1. Crear una Cuenta en el Servicio de Alojamiento:
2. Regístrate en el servicio de alojamiento elegido.
3. Subir Archivos del Proyecto:
4. Accede al panel de control de tu cuenta y busca la opción para cargar archivos (usualmente "File Manager").
5. Carga todos los archivos de la carpeta del proyecto SAAR en el directorio admin_html o equivalente.

- **Crear la Base de Datos:**

1. En el panel de control, busca la opción de bases de datos (MySQL) y crea una nueva base de datos.
2. Anota el nombre de la base de datos, el nombre de usuario y la contraseña.

- **Importar la Base de Datos:**

1. Accede a phpMyAdmin (si está disponible en el panel de control).
2. Crea una nueva base de datos y usa la opción "Importar" para cargar el archivo db_taller_rodriguez.sql.

- **Configurar Conexión a la Base de Datos:**

1. Abre el archivo api/helpers/config.php y modifica las credenciales de conexión con los datos de la base de datos creada en el servicio de alojamiento.

- **Acceder al Proyecto:**

1. Usa la URL proporcionada por el servicio de alojamiento para acceder al sitio privado (ejemplo: http://tu_dominio.com/views/admin/)

- **Crear un Usuario:**

1. Al ingresar al sitio privado por primera vez, se te pedirá que crees un usuario administrador.