**Diccionario de datos**

**Tablas:**

|  |  |  |
| --- | --- | --- |
| **Usuarios** | | |
| **Campo** | **Descripción** | **Tipo de dato** |
| Id (clave primaria) | Identificador único del usuario. Es un valor numérico que se genera automáticamente y se incrementa con cada nuevo registro. | Int(11) |
| Nombre\_usuario | Nombre de usuario que se utiliza para iniciar sesión en el sistema. Es una cadena de texto que identifica al usuario. | Varchar(50) |
| contrasenia | Contraseña del usuario. Es una cadena de texto que se almacena de forma segura, ya que es la clave para acceder al sistema. | Varchar(25) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Registro\_usuarios** | | | |
| **Campo** | **Descripción** | **Tipo de dato** | **Observación** |
| ID (clave primaria) | Identificador único del registro. Es un valor numérico que se genera automáticamente y se incrementa con cada nuevo registro. | Int(11) | * Esta tabla se utiliza para almacenar información adicional sobre los usuarios registrados, como su sector, número de teléfono, correo electrónico y número de lote. * La relación con la tabla "usuarios" se establece mediante la llave foránea  id\_usuario, que permite vincular cada registro a un usuario específico. * La información almacenada en esta tabla se puede utilizar para mejorar la comunicación con los usuarios, identificar su ubicación y ofrecer servicios personalizados. |
| id\_usuario (*clave foránea que depende de la tabla usuarios*) | Identificador del usuario asociado a este registro. Es una llave foránea que hace referencia a la tabla "usuarios" y permite asociar un registro a un usuario específico. | Int(11) |
| sector | Sector al que pertenece el usuario. Es una cadena de texto que indica el sector o zona geográfica donde reside el usuario. | Varchar(200) |
| teléfono | Número de teléfono del usuario. Es una cadena de texto que contiene el número de teléfono del usuario. | Varchar(30) |
| correo\_electronico | Correo electrónico del usuario. Es una cadena de texto que contiene la dirección de correo electrónico del usuario. | Varchar(40) |
| lote | Número de lote del usuario. Es una cadena de texto que indica el número de lote o parcela de tierra donde reside el usuario. | Varchar(400) |

|  |  |  |
| --- | --- | --- |
| **Contribuyentes** | | |
| **Campo** | **Descripción** | **Tipo de dato** |
| Id (Clave primaria) | Identificador único del contribuyente. Es un valor numérico que se genera automáticamente y se incrementa con cada nuevo registro. | Int(11) |
| id\_usuario(*clave foránea que depende de la tabla usuarios*) | Identificador del usuario asociado a este contribuyente. Es una llave foránea que hace referencia a la tabla "usuarios" y permite asociar a un usuario con un contribuyente. | Int(11) |
| nombre | Nombre completo del contribuyente. Es una cadena de texto que almacena el nombre del contribuyente. | Varchar(100) |
| apellido | Apellido completo del contribuyente. Es una cadena de texto que almacena el apellido del contribuyente. | Varchar(100) |
| Dirección | Dirección completa del contribuyente. Es una cadena de texto que contiene la dirección física del contribuyente. | Varchar(255) |
| Teléfono | Número de teléfono del contribuyente. Es una cadena de texto que contiene el número de teléfono del contribuyente. | Varchar(20 |
| Correo\_electronico | Correo electrónico del contribuyente. Es una cadena de texto que contiene la dirección de correo electrónico del contribuyente. | Varchar(40) |
| estado | Estado del contribuyente. Es un valor de tipo enumerado que indica si el contribuyente está activo o inactivo. | Enum(activo, inactivo) |

|  |  |  |
| --- | --- | --- |
| **Pagos** | | |
| **Campo** | **Descripción** | **Tipo de dato** |
| Id (clave primaria) | Identificador único del pago. Es un valor numérico que se genera automáticamente y se incrementa con cada nuevo registro. | Int(11) |
| Id\_contribuyente (*clave foránea que depende de la tabla contribuyentes)* | Identificador del contribuyente que realizó el pago. Es una llave foránea que hace referencia a la tabla "contribuyentes" y permite asociar un pago a un contribuyente. | Int(11) |
| Monto\_pago | Monto del pago realizado. Es un valor de tipo decimal que representa el monto del pago. | Decimal(10,2) |
| Fecha\_pago | Fecha en que se realizó el pago. Es un valor de tipo fecha que indica la fecha en que se realizó el pago. | date |
| Periodo\_pago (mes, trimestre, semestre, anual) | Periodo del pago (por ejemplo, mes, trimestre, año). Es un valor de tipo enumerado que indica el periodo al que corresponde el pago. | Enum(mes, trimestre, semestre, anual) |
| Fecha\_vencimiento | Fecha de vencimiento del pago. Es un valor de tipo fecha que indica la fecha límite para realizar el pago. | date |
| Id\_tarifa (*clave foránea que depende de la tabla tarifas)* | Identificador de la tarifa aplicada al pago. Es una llave foránea que hace referencia a la tabla "tarifas" y permite asociar un pago a una tarifa específica. | Int(11) |

|  |  |  |
| --- | --- | --- |
| **Tarifas** | | |
| **Campo** | **Descripción** | **Tipo de dato** |
| Id (clave primaria) | Identificador único de la tarifa. Es un valor numérico que se genera automáticamente y se incrementa con cada nuevo registro. | Int(11) |
| Descripción | Descripción de la tarifa. Es una cadena de texto que describe el tipo de tarifa. | Varchar(255) |
| Monto\_tarifa | Monto de la tarifa. Es un valor de tipo decimal que representa el monto de la tarifa. | Decimal(10,2) |
| Periodo\_pago (mes, trimestre, semestre, anual) | Periodo de la tarifa (por ejemplo, "mensual", "trimestral", "anual"). Es un valor de tipo enumerado que indica el periodo de la tarifa. | Enum(mes, trimestre, semestre, anual) |

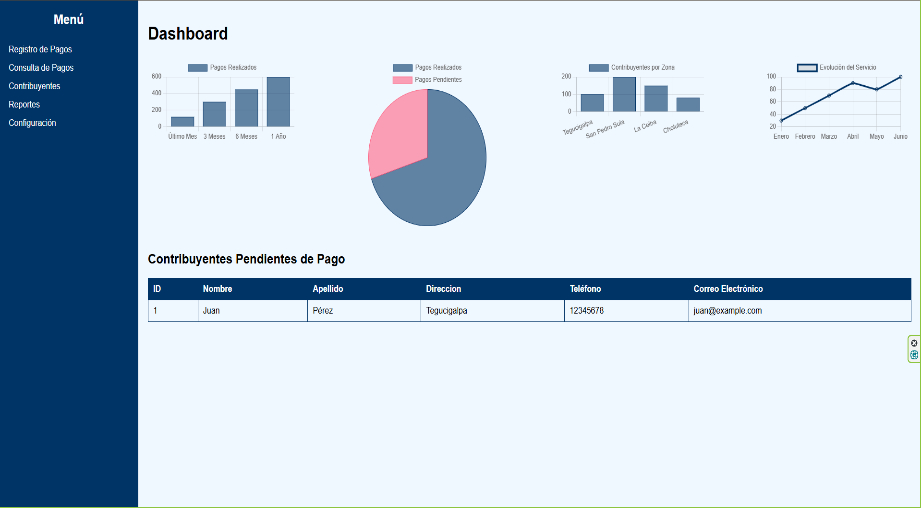
**Prototipo**

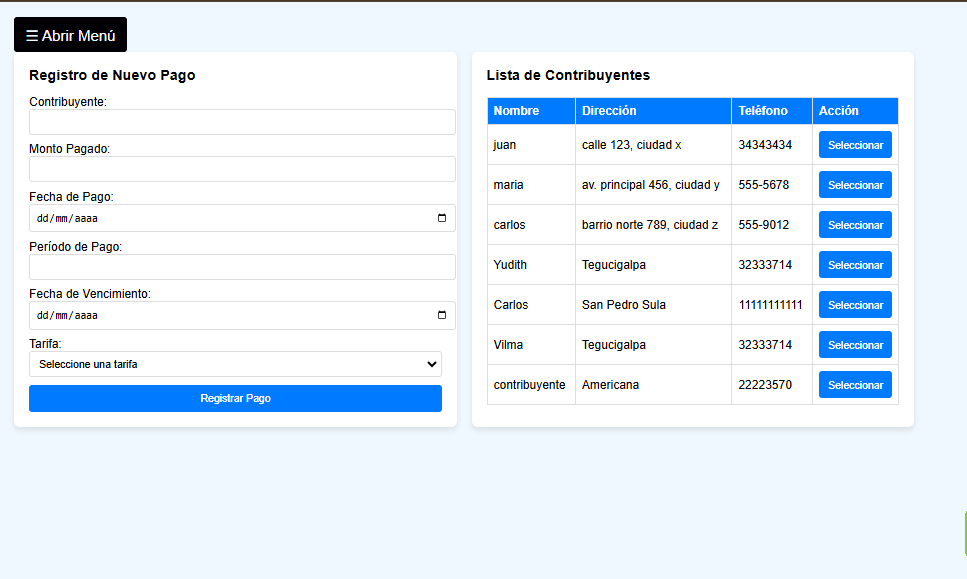
**Pantallas:**

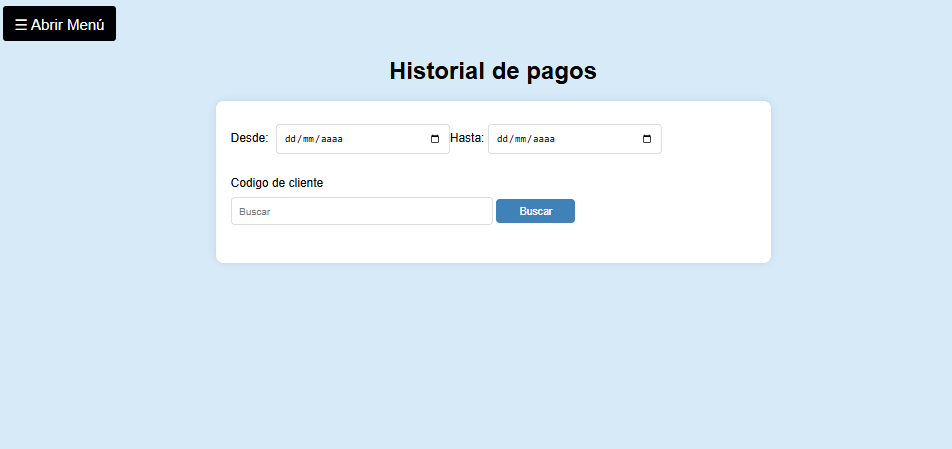
1. **Login**

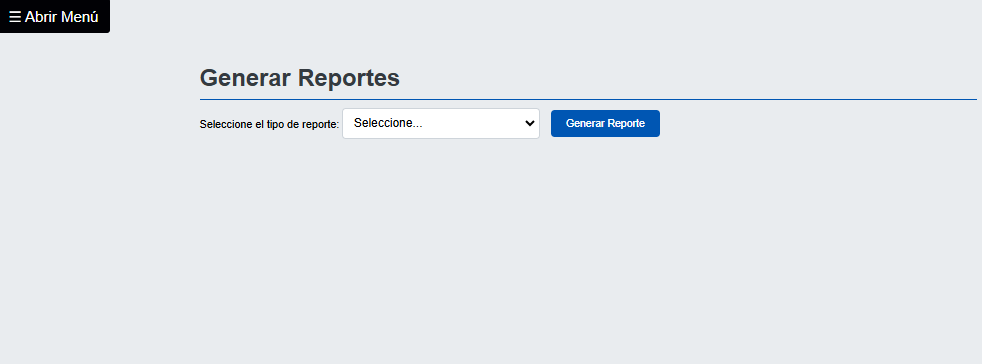


1. **Inicio-Dashboard**

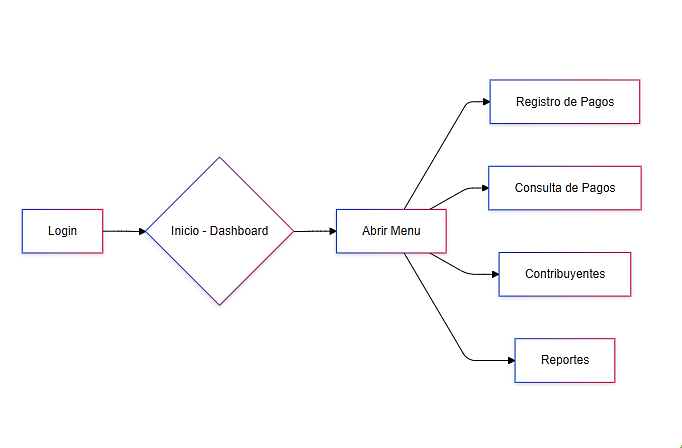


1. **Registro de pagos**
2. **Consulta de pagos**



1. **Contribuyentes**
2. **reportes**

**Diagrama descriptivo de la aplicación web:**

****

**Diseño de interfaz:**

**Paleta de Colores utilizada:**

La aplicación web utiliza una paleta de colores que combina tonos de azul, gris y blanco, creando una interfaz más limpia.

**A continuación, se detallan los colores utilizados:**

* Fondo del cuerpo: #f0f8ff (azul claro)
* Fondo del contenedor: #ffffff (blanco)
* Fondo del menú lateral: #111111 (casi negro) y #343a40 (gris oscuro)
* Texto principal: #333333 (gris oscuro)
* Texto de encabezados: #007BFF (azul)
* Colores de botones: #007BFF (azul) con hover en #0056b3 (azul más oscuro)

**Tipografía:**

La tipografía utilizada en la aplicación es Arial, sans-serif, lo que proporciona una apariencia moderna y legible. Los tamaños de fuente varían según el elemento, siendo 14px el tamaño estándar para botones e inputs, mientras que los encabezados (h1, h2, h3) tienen tamaños más grandes, con h1 establecido en 2.2em.

**Interacción del usuario en la aplicación web:**

1. **Pantalla de Login:**

**Nombre: Login**

Elementos:

* Campo de texto para "Nombre de usuario".
* Campo de texto para "Contraseña".
* Botón "Iniciar sesión".
* Enlace "Recuperar contraseña".
* Enlace "Registrar un nuevo usuario".

Flujo:

* Si las credenciales son válidas, redirige al Inicio.
* Si las credenciales son inválidas, muestra un mensaje de error.

**2. Pantalla de Inicio:**

**Nombre: Dashboard**

Elementos:

* Gráfico 1: Resumen de Pagos Realizados (último mes, 3 meses, 6 meses, un año).
* Gráfico 2: Evolución del Servicio (fechas de pagos realizados).
* Gráfico 3: Porcentaje de Pagos (círculo que muestra pagos realizados y pendientes).
* Gráfico 4: Contribuyentes por Zona (muestra la ubicación geográfica de los contribuyentes activos).
* Tabla: Lista de contribuyentes con pagos pendientes.
* Mapa: Mapa interactivo que permite buscar zonas geográficamente.

Menú Desplegable: "Abrir menú" con las opciones:

* Registro de Pagos
* Consulta de Pagos
* Contribuyentes
* Reportes

**3**. **Pantalla Registro de Pagos:**

**Nombre: Registro de Pagos**

Elementos:

Tabla de Contribuyentes Activos: Muestra la lista de contribuyentes activos.

* Selección de Contribuyente: Permite seleccionar un contribuyente de la tabla.

Formulario de Pago:

Campos:

* Contribuyente (se rellena automáticamente al seleccionar al contribuyente).
* Monto Pagado.
* Fecha de Pago.
* Tarifa Aplicada (selección de una tarifa).
* Botón "Registrar Pago": Realiza el registro del pago.

**4. Pantalla Consulta de Pagos:**

**Nombre: Historial de Pagos**

Elementos:

* Filtro por Fecha: Campo para ingresar un rango de fechas (inicio y fin) o una fecha específica.
* Filtro por ID de Contribuyente: Campo para ingresar el ID del contribuyente.
* Tabla de Pagos: Muestra la lista de pagos que coinciden con los filtros seleccionados.
* Información: Se muestra la información pertinente al pago

Flujo:

* El usuario ingresa los filtros y presiona "Buscar".
* La aplicación muestra los resultados que coinciden con los filtros.

**5. Pantalla Contribuyentes:**

**Nombre: Administrar Contribuyentes**

Elementos:

* Tabla de Contribuyentes: Muestra la información de los contribuyentes.
* Información: ID, Nombre, Apellido, Dirección, Teléfono, Correo electrónico.

Acciones:

* Editar: Abre un formulario para modificar los datos del contribuyente seleccionado.
* Habilitar/Deshabilitar: Activa o desactiva el contribuyente en el sistema.
* Botón "Agregar Contribuyente": Abre un formulario para agregar un nuevo contribuyente.
* Formulario Agregar Contribuyente:
* Campos: Nombre, Apellido, Dirección, Teléfono, Correo electrónico.

**6. Pantalla Reportes:**

**Nombre: Generar Reportes**

Elementos:

Opciones de Reportes:

* Contribuyentes: Muestra un reporte con la información de los contribuyentes.
* Pagos: Muestra un reporte con la información de los pagos realizados.
* Tarifas: Muestra un reporte con la información de las tarifas.
* Pagos Pendientes: Muestra un reporte con la información de los pagos pendientes.
* Opciones de Exportación: Permite exportar los reportes a un formato PDF.

**Implementación del sistema:**

* **Frontend:** HTML, CSS, JavaScript
* **Backend:** PHP y MySQL
* **Editor de código:** Visual Studio Code
* **Servidor web local:** XAMPP
* **Herramienta para administrar la base de datos:** PHPMyAdmin

**Bibliotecas:**

* **Chart.js**: Biblioteca de JavaScript que se utiliza para crear gráficos interactivos.
* **OpenStreetMap:** iframe que muestra un mapa de OpenStreetMap, lo que permite visualizar la zona geográfica relacionada con los datos presentados en el dashboard.
* **SweetAlert:** Sirve para mostrar mensajes de éxito o error
* **html2canvas:** Captura el contenido HTML y lo convierte en una imagen.
* **jsPDF:** Genera documentos PDF a partir de contenido HTML.

**Diseño y Desarrollo del código:**

**Creación de la base de datos:**

**Relaciones entre tablas:**

* **Usuarios 🡨 🡪 registro\_usuarios** **(1:1)** cada usuario tiene un único registro en la tabla registro\_usuarios
* **Usuarios 🡪 contribuyentes (1:0..1)** un usuario puede ser o no ser un contribuyente y un contribuyente pertenece a un usuario
* **Contribuyentes 🡪 pagos (1:M)** un contribuyente puede realizar varios pagos
* **Tarifas🡪 pagos (1:M)** una tarifa puede aplicarse a varios pagos

**Integración Frontend-Backend:**

**INICIO-DASHBOARD:**

* **Interacción del Usuario:** El usuario accede al dashboard y se visualizan gráficos y una tabla que muestran información sobre los pagos y contribuyentes.
* **Solicitudes al Backend:** Cuando se carga la página, el frontend envía solicitudes al backend para obtener datos específicos. Esto se realiza mediante la función fetch en JavaScript, que hace solicitudes HTTP al archivo PHP.
* **Procesamiento en el Backend:** El backend recibe las solicitudes, ejecuta las consultas SQL necesarias para obtener los datos requeridos y devuelve los resultados en formato JSON.
* **Respuesta al Frontend:** El frontend recibe la respuesta JSON y utiliza los datos para actualizar la interfaz de usuario, mostrando los gráficos y los datos de la tabla con la información solicitada.
* **Actualización de la Interfaz:** Los gráficos y tabla se actualizan dinámicamente en la página sin necesidad de recargarla, gracias a la manipulación del DOM en JavaScript.

**REGISTRO DE PAGOS:**

* **Interacción del Usuario:** El usuario selecciona un contribuyente y una tarifa en el frontend.
* **Solicitudes al Backend:** Cuando se selecciona un contribuyente o una tarifa, el frontend envía solicitudes GET al backend para obtener los datos necesarios. Al enviar el formulario de pago, se envía una solicitud POST al backend para guardar el registro.
* **Procesamiento en el Backend:** El backend procesa las solicitudes GET y POST, realiza las operaciones necesarias (consultar datos o guardar registros) y devuelve las respuestas al frontend.
* **Respuesta al Frontend:** El frontend recibe las respuestas del backend y actualiza la interfaz de usuario en función de los datos recibidos.
* **Actualización de la Interfaz:** El frontend muestra notificaciones de éxito o error en función de la respuesta del backend.

**CONSULTA DE PAGOS:**

* **Interacción del Usuario:**
* El usuario abre la página Historial de pagos
* El usuario ingresa un rango de fechas en los campos "Desde" y "Hasta".
* El usuario ingresa un código de contribuyente en el campo de búsqueda.
* El usuario hace clic en el botón "Buscar".
* **Captura de Datos en el Frontend:**

Al hacer clic en el botón "Buscar", se activa un evento en JavaScript que captura los valores de los campos de entrada:

**desde:** Valor del campo de fecha "Desde".

**hasta:** Valor del campo de fecha "Hasta".

**Código Contribuyente:** Valor del campo de búsqueda.

* **Solicitud al Backend:** El frontend utiliza la función fetch para enviar una solicitud HTTP GET al archivo que se maneja para buscar pago
* VerifyEditCopy code
* **Procesamiento en el Backend:** El archivo para buscar pago recibe la solicitud y extrae los parámetros desde, hasta y código.
* Se establece una conexión a la base de datos utilizando **conexion.php.**
* Se construye una consulta SQL para buscar los pagos que coincidan con los criterios proporcionados.
* Se ejecuta la consulta y se obtienen los resultados.
* **Devolución de Resultados:**
* Los resultados de la consulta se almacenan en un array y se convierten a formato JSON.
* El backend envía la respuesta JSON de vuelta al frontend.
* **Recepción de Datos en el Frontend:**

El frontend recibe la respuesta JSON.

* **Se procesa la respuesta:**
* Si hay resultados, se crean elementos HTML para mostrar cada pago y elementos pertinentes.
* Si no hay resultados, se muestra un mensaje indicando que no se encontraron pagos.
* **Visualización de Resultados:**

Los resultados se muestran dinámicamente en la página sin necesidad de recargarla, gracias a la manipulación del DOM en JavaScript.

**CONTRIBUYENTES:**

* **Interacción del Usuario:**
* El usuario hace clic en el botón "AGREGAR CONTRIBUYENTE", lo que abre un modal con un formulario.
* El usuario completa el formulario y lo envía.
* **Manejo del Formulario:**
* Al enviar el formulario, se previene el comportamiento predeterminado (recarga de página) y se envían los datos a lib/procesar.php mediante una solicitud AJAX (usando XMLHttpRequest).
* El servidor procesa los datos y devuelve una respuesta en formato JSON.
* **Respuesta del Servidor:**
* Si la operación es exitosa, se muestra un mensaje de éxito usando SweetAlert2 y se recarga la página para reflejar los cambios.
* Si hay un error, se muestra un mensaje de error.
* **Edición de Contribuyentes:**
* Al hacer clic en "EDITAR", se realiza una solicitud AJAX a lib/obtener\_contribuyente.php para obtener los datos del contribuyente seleccionado.
* Los datos se cargan en el formulario del modal, que se muestra al usuario.
* **Habilitar/Deshabilitar Contribuyentes:**
* Al hacer clic en "DESHABILITAR" o "HABILITAR", se muestra un cuadro de confirmación.
* Si el usuario confirma, se envía una solicitud AJAX a lib/eliminar\_contribuyente.php o lib/habilitar\_contribuyente.php para actualizar el estado del contribuyente en la base de datos.
* Se muestra un mensaje de éxito y se recarga la página.

**REPORTES:**

* **Interacción del Usuario:**

El usuario selecciona un tipo de reporte en un formulario y hace clic en el botón "Generar Reporte".

* **Manejo del Formulario:**

Al enviar el formulario, se previene el comportamiento predeterminado y se redirige al usuario a lib/generar\_reporte.php con el tipo de reporte seleccionado como parámetro en la URL.

* **Generación del Reporte en el Backend:**
* En lib/generar\_reporte.php, se recibe el tipo de reporte y se ejecuta una consulta SQL correspondiente para obtener los datos.
* Los datos se almacenan en un array y se preparan para ser mostrados en una tabla HTML.
* **Visualización del Reporte:**
* Se genera una tabla HTML con los datos obtenidos y se muestra al usuario.
* Se proporciona un botón para descargar el reporte en formato PDF.
* **Generación del PDF:**

Al hacer clic en el botón "Descargar PDF", se utiliza html2canvas y jsPDF para capturar la tabla y generar un archivo PDF que se descarga automáticamente.

**Conexión con la base de datos: archivo: conexion.php**

**Captura de pantalla de un celular

Descripción generada automáticamente**

**Lógica utilizada en las consultas SQL en las diferentes pantallas de la aplicación web:**

**DASHBOARD:**

Obtener la suma de los montos de los pagos realizados en diferentes periodos de tiempo (último mes, tres meses, seis meses y un año).

Uso de SUM y CASE: Se utiliza la función SUM junto con CASE para calcular la suma de monto\_pago solo para aquellos registros que cumplen con las condiciones de fecha especificadas.

Intervalos de Tiempo: Se utilizan funciones de fecha de MySQL (DATE\_SUB y CURDATE()) para definir los intervalos de tiempo.

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

Calcular el porcentaje de contribuyentes que han realizado pagos y el porcentaje de contribuyentes que tienen pagos pendientes.

Cálculo de Porcentajes: Se cuenta el número de contribuyentes únicos que han realizado pagos y se divide por el total de contribuyentes activos (estado = 1).

Condiciones: Solo se cuentan los pagos cuya fecha de vencimiento es mayor o igual a la fecha actual.

**Texto

Descripción generada automáticamente**

Obtener la cantidad de contribuyentes agrupados por zona geográfica.

Uso de SUBSTRING\_INDEX: Se utiliza para extraer la zona de la dirección, asumiendo que la dirección está separada por comas.

Agrupación: Se agrupan los resultados por zona para contar cuántos contribuyentes hay en cada una.

**Texto

Descripción generada automáticamente**

Obtener la evolución de los pagos a lo largo del tiempo.

Agrupación por Fecha: Se agrupan los pagos por fecha\_pago y se suman los montos para cada fecha.

Ordenación: Se ordenan los resultados por fecha para mostrar la evolución cronológica.

**Texto

Descripción generada automáticamente**

Obtener una lista de contribuyentes que tienen pagos pendientes.

Uso de LEFT JOIN: Se utiliza un LEFT JOIN para combinar la tabla de contribuyentes con la tabla de pagos. Esto permite obtener todos los contribuyentes, incluso aquellos que no tienen registros de pagos (es decir, que no han realizado ningún pago).

**Condiciones:**

c.estado = 1: Solo se consideran contribuyentes activos.

(p.fecha\_vencimiento < CURDATE() OR p.fecha\_vencimiento IS NULL): Se filtran los contribuyentes que tienen pagos vencidos (fecha de vencimiento anterior a la fecha actual) o que no tienen pagos registrados (es decir, fecha\_vencimiento es NULL).

Uso de DISTINCT: Se utiliza para asegurarse de que no haya duplicados en los resultados, en caso de que un contribuyente tenga múltiples pagos.

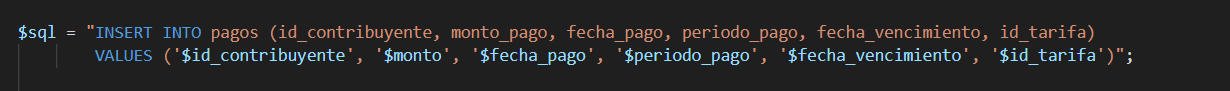
Límite de Resultados: Se limita la cantidad de resultados a 25 para evitar que la consulta devuelva demasiados registros a la vez.

**Texto

Descripción generada automáticamente**

**REGISTRO DE PAGOS:**

El objetivo de esta consulta es registrar un nuevo pago en la base de datos, asociando el pago a un contribuyente específico y proporcionando detalles sobre el monto, la fecha y el periodo del pago, así como la fecha de vencimiento y la tarifa correspondiente.

****

**CONSULTA DE PAGOS:**

**Ingresar aquí la consulta SQL realizada para esta pantalla en el archivo PHP actualizado.**

**CONTRIBUYENTES:**

Conexión: Se establece la conexión a la base de datos.

Verificación de Conexión: Se verifica si la conexión fue exitosa.

Obtener ID: Se obtiene el ID del contribuyente a deshabilitar a través de una solicitud GET.

Consulta SQL: Se ejecuta una consulta UPDATE para cambiar el estado del contribuyente a "inactivo".

Resultado: Se devuelve un mensaje de éxito o error según el resultado de la consulta.

**Texto

Descripción generada automáticamente**

Tipo de Consulta: UPDATE - Similar a la anterior, se utiliza para modificar registros.

Tabla: contribuyentes - La misma tabla.

Columna a Modificar: estado - Se cambia el estado del contribuyente a "activo".

Condición: WHERE id = $id - Se actualiza el registro del contribuyente específico.

**Texto

Descripción generada automáticamente**

Tipo de Consulta: SELECT - Se utiliza para recuperar datos de la base de datos.

Tabla: contribuyentes - La tabla de donde se obtienen los datos.

Condición: WHERE id = $id - Se selecciona el registro donde el id coincide con el proporcionado. Esto permite obtener la información de un contribuyente específico.

**Texto

Descripción generada automáticamente**

Tipo de Consulta: Puede ser UPDATE o INSERT, dependiendo de si se proporciona un id o no.

Tabla: contribuyentes - La tabla donde se almacenan los datos.

Actualización:

Columnas a Modificar: Se actualizan las columnas nombre, apellido, direccion, telefono, y correo\_electronico.

Condición: WHERE id=$id - Solo se actualiza el registro del contribuyente específico.

Inserción:

Columnas a Insertar: Se insertan los valores en las columnas correspondientes.

Valores: Se utilizan las variables proporcionadas en el formulario.

**Captura de pantalla con la imagen de una pantalla

Descripción generada automáticamente**

**REPORTES:**

Obtener todos los contribuyentes que están activos (estado = 1).

Tabla: contribuyentes - Esta tabla contiene la información de los contribuyentes.

Condición: WHERE estado = 1 - Filtra los registros para incluir solo aquellos que están activos.

Ordenamiento: ORDER BY id ASC - Los resultados se ordenan por el id de manera ascendente.

****

Obtener todos los registros de pagos.

Tabla: pagos - Esta tabla contiene la información sobre los pagos realizados.

Ordenamiento: ORDER BY id ASC - Los resultados se ordenan por el id de manera ascendente.

****

Obtener información sobre las tarifas disponibles.

Tabla: tarifas - Esta tabla contiene las tarifas que se aplican a los contribuyentes.

Columnas Seleccionadas: id, descripcion, monto\_tarifa - Solo se seleccionan estas columnas específicas.

Ordenamiento: ORDER BY id ASC - Los resultados se ordenan por el id de manera ascendente.

****

Obtener una lista de contribuyentes que tienen pagos pendientes.

Tablas:

contribuyentes c - Se utiliza para obtener la información de los contribuyentes.

pagos p - Se utiliza para obtener la información de los pagos.

JOIN: LEFT JOIN - Se realiza una unión izquierda entre las tablas contribuyentes y pagos para incluir todos los contribuyentes, incluso aquellos que no tienen pagos registrados.

Condiciones:

c.estado = 1 - Solo se incluyen contribuyentes activos.

(p.fecha\_vencimiento < CURDATE() OR p.fecha\_vencimiento IS NULL) - Se filtran los pagos cuya fecha de vencimiento es anterior a la fecha actual o que no tienen fecha de vencimiento (lo que indica que no se ha registrado un pago).

Ordenamiento: ORDER BY p.fecha\_vencimiento ASC - Los resultados se ordenan por la fecha de vencimiento de los pagos de manera ascendente.

Límite: LIMIT 0, 50 - Se limita el número de resultados a 50.

**Texto

Descripción generada automáticamente**

**Pruebas realizadas:**

Las pruebas realizadas durante la implementación de la aplicación web para la Junta Administradora de Agua Potable y Saneamiento fueron exhaustivas y exitosas. Se realizaron pruebas unitarias para cada módulo del código PHP, asegurando que cada función funcionaba correctamente de forma independiente.

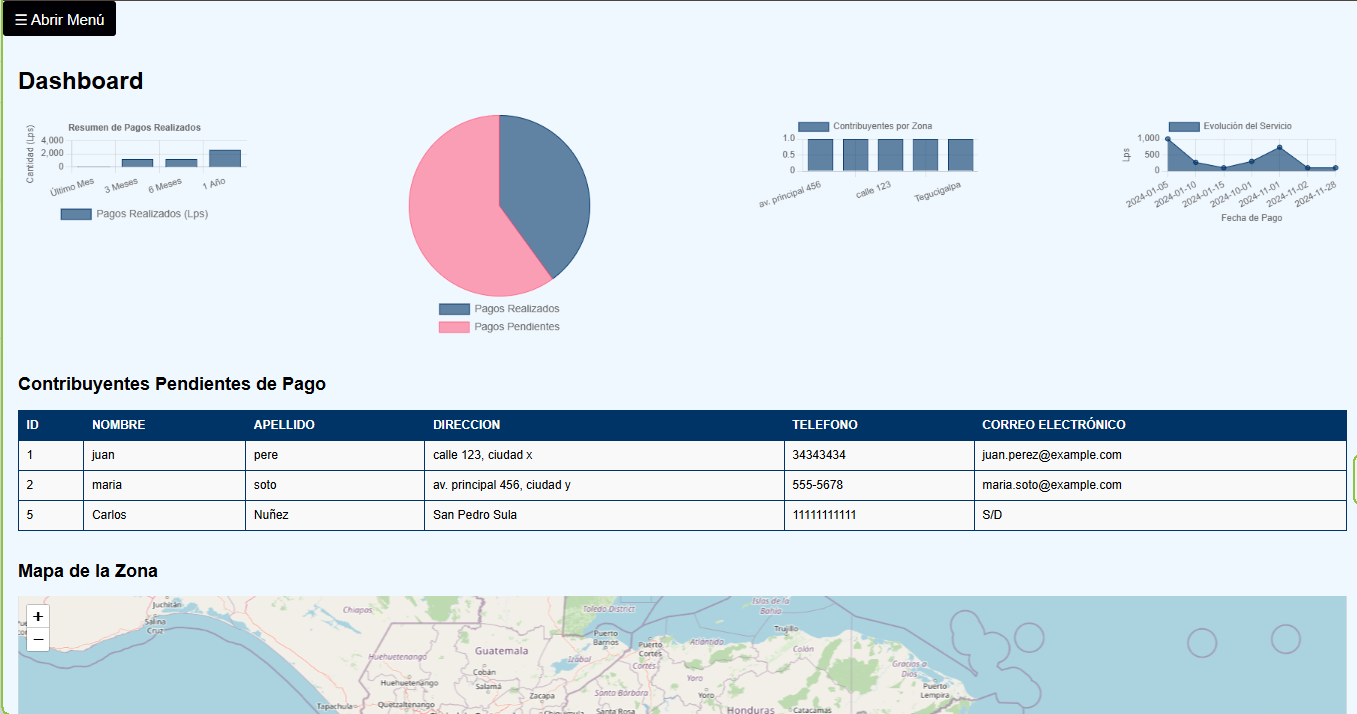
También se llevaron a cabo pruebas de integración para verificar la comunicación entre el frontend y el backend, así como la interacción con la base de datos MySQL, y las pruebas de aceptación con datos simulados confirmaron que la aplicación cumplía con los requisitos funcionales. No se encontraron errores significativos durante las pruebas, y los errores menores que se detectaron se corrigieron con éxito.

Para optimizar el rendimiento de la aplicación, se optimizaron las consultas SQL. Además, se tomaron algunas medidas de seguridad para proteger la aplicación de ataques, como la validación de entradas, la encriptación de contraseñas, la protección contra inyecciones SQL y la implementación de HTTPS."

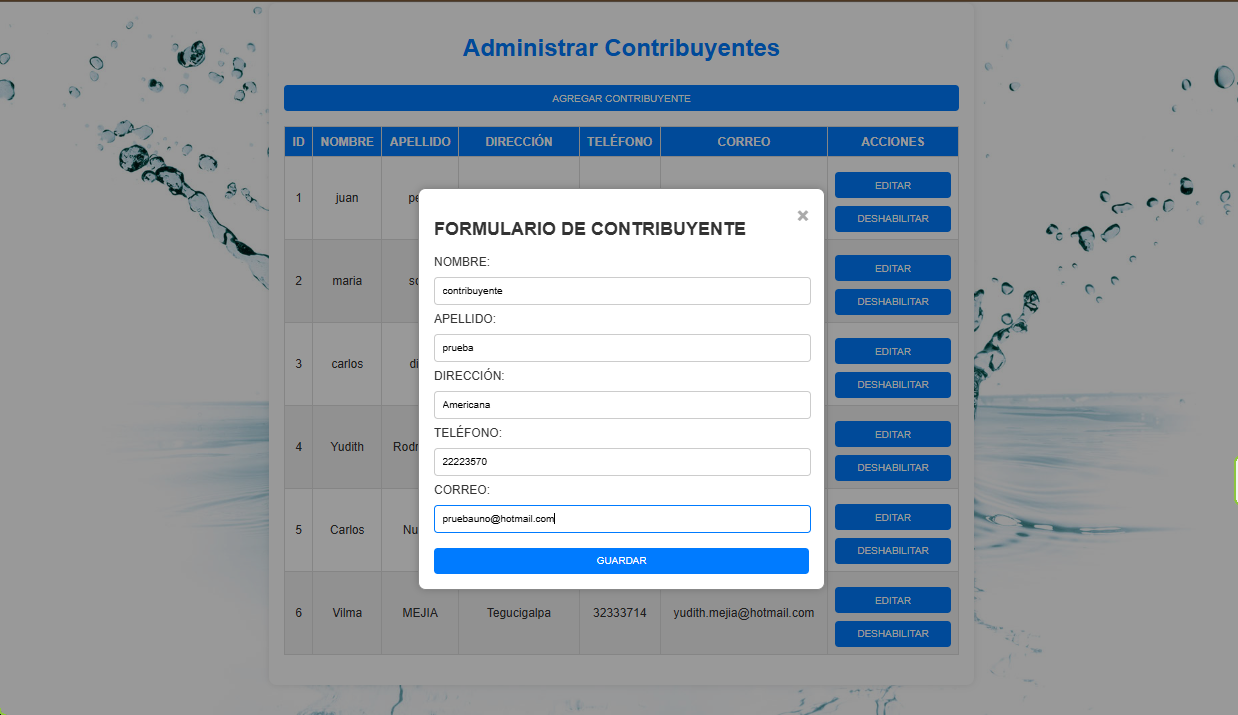
**Ejecución del sistema con datos de prueba:**

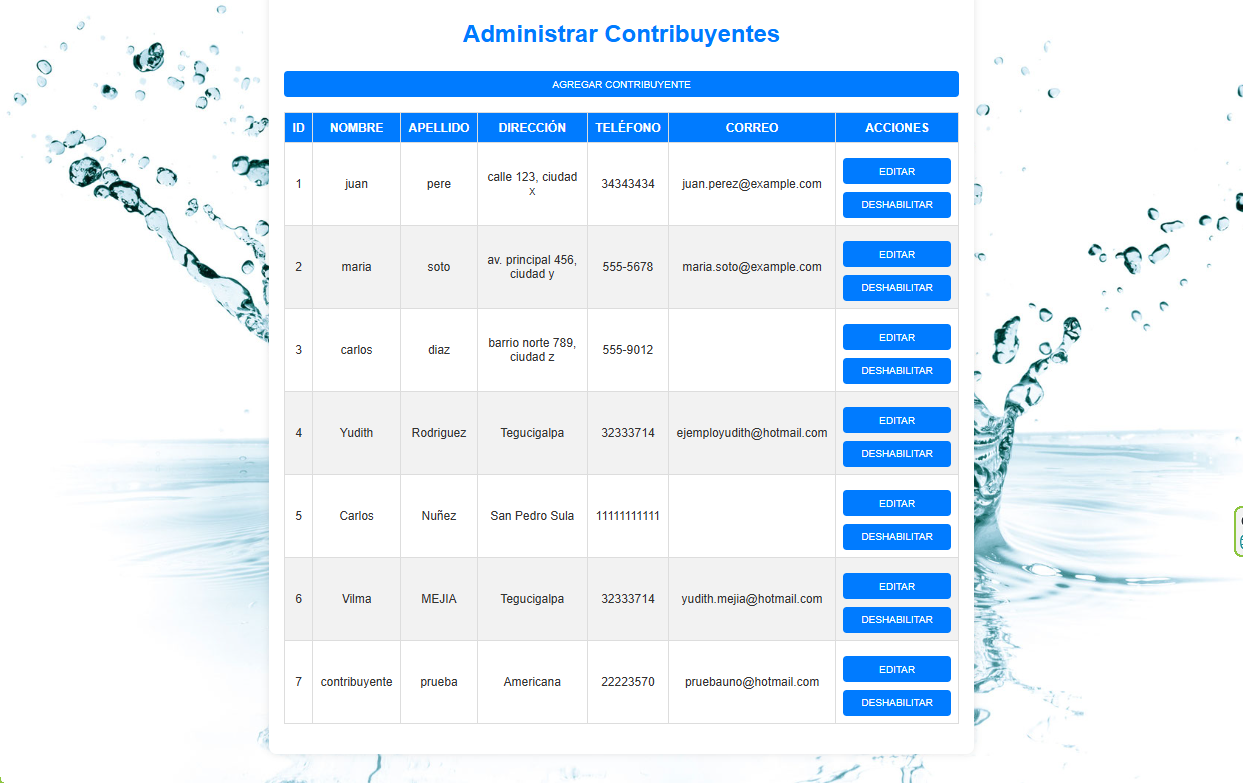
Interfaz de usuario gráfica, Aplicación

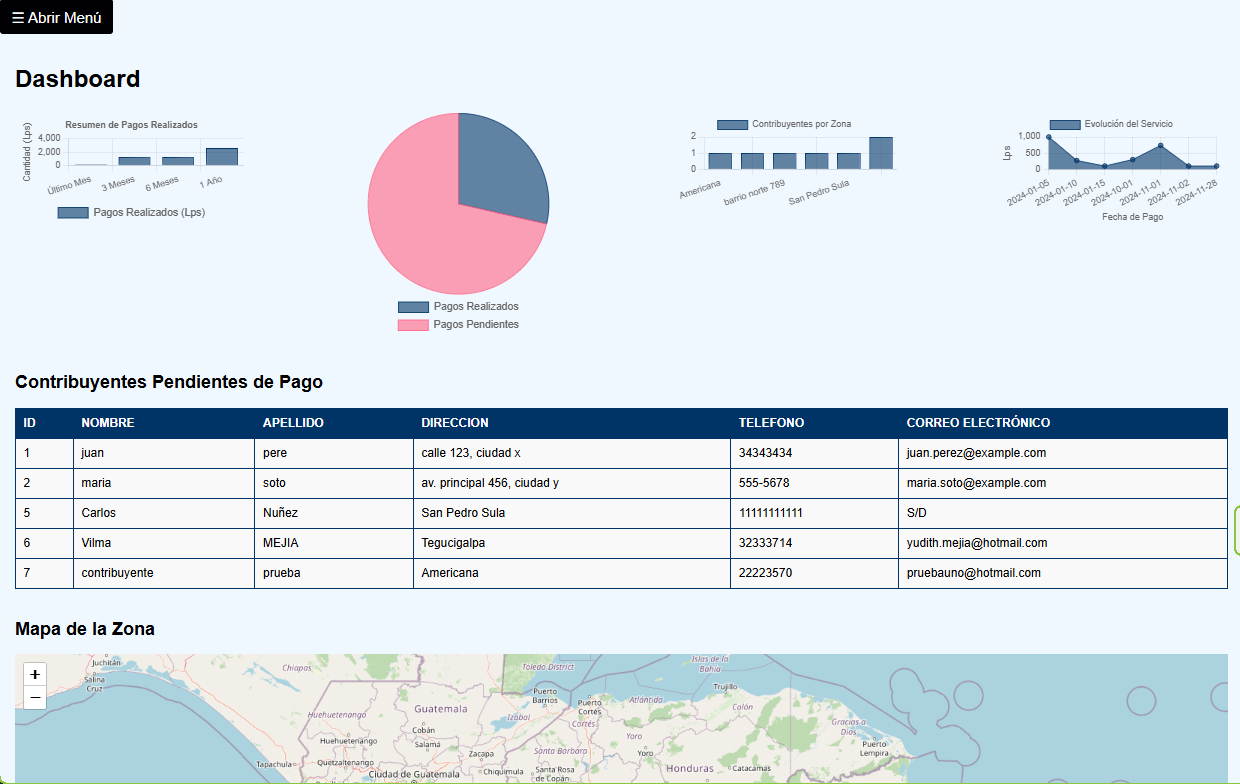
Descripción generada automáticamente

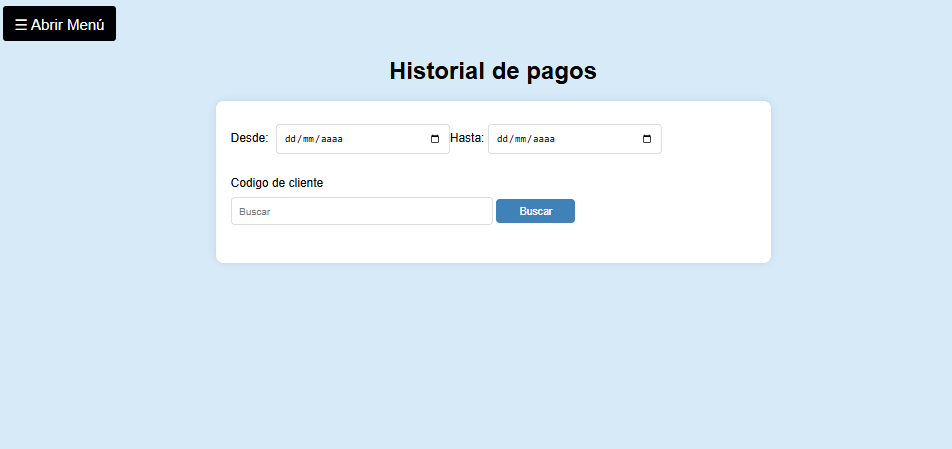












Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente