Tabla: users

La tabla **users** almacena a todas las personas que interactúan con el sistema, independientemente de su rol. Esta tabla es el punto de partida para la autenticación, autorización y trazabilidad de acciones dentro de la plataforma.

Se decidió unificar a conductores, proveedores y administradores en una sola tabla para simplificar la gestión de identidad, evitando duplicación de datos y facilitando la implementación de controles de acceso. El campo de rol permite definir el comportamiento del usuario dentro del sistema, mientras que el estatus controla su acceso operativo.

Tabla: user_consents

Esta tabla existe para registrar la aceptación de términos legales, políticas de privacidad y otros consentimientos requeridos para operar una plataforma digital real. No basta con asumir que un usuario aceptó condiciones; es necesario registrar cuándo, cómo y bajo qué versión.

Esto protege tanto al usuario como a la plataforma, y permite cumplir con buenas prácticas legales y de protección de datos, especialmente cuando se manejan datos sensibles como ubicación y transacciones.

Tabla: service_areas

La tabla **service_areas** define las zonas geográficas donde el sistema opera. En un proyecto de asistencia vial, no es realista asumir cobertura total desde el inicio. Esta tabla permite delimitar ciudades o regiones activas y evita que se generen solicitudes en zonas no atendidas.

Además, facilita la expansión futura del proyecto a nuevas ciudades sin modificar la lógica central del sistema.

Tabla: area_polygons

Esta tabla complementa a **service_areas** y permite definir con mayor precisión la cobertura mediante polígonos geográficos. Aunque puede parecer avanzada, su inclusión permite implementar validaciones geográficas más exactas y prepara al sistema para un crecimiento real.

Tabla: providers

La tabla **providers** almacena información específica de los prestadores de servicio. Aunque un proveedor es un usuario, tiene características adicionales como validación, zona de operación y estatus operativo, que no aplican a otros roles.

Separar esta información permite controlar mejor la operación y evita sobrecargar la tabla de usuarios con datos que no les corresponden a todos.

Tabla: provider_services

No todos los proveedores ofrecen los mismos servicios. Esta tabla permite definir qué servicios puede atender cada proveedor, resolviendo una relación muchos a muchos entre proveedores y servicios.

Esto es fundamental para asignar correctamente las solicitudes y evitar que un proveedor reciba un servicio que no puede atender.

Tabla: provider_schedules

La disponibilidad de un proveedor no es constante. La tabla **provider_schedules** permite definir horarios de atención, haciendo el sistema más realista y evitando asignaciones fuera del horario operativo del proveedor.

Esto también facilita el cálculo de tiempos de respuesta y mejora la experiencia del usuario.

Tabla: provider_locations

La tabla **provider_locations** registra la ubicación del proveedor, permitiendo asignaciones basadas en cercanía. No se requiere tracking en tiempo real avanzado, pero sí un registro mínimo que permita decisiones operativas inteligentes.

Además, este historial puede ser útil para análisis de cobertura y rendimiento.

Tabla: provider_documents

En un servicio que implica interacción física con el usuario, es indispensable validar a los proveedores. Esta tabla permite almacenar documentos como identificaciones, licencias o comprobantes, junto con su estatus de validación.

Esto fortalece la confianza en la plataforma y permite cumplir con estándares mínimos de seguridad.

Tabla: vehicles

La tabla **vehicles** almacena información básica de los vehículos de los conductores. Esta información ayuda a los proveedores a llegar preparados y evita errores durante la atención del servicio.

Se decidió mantenerla como una entidad separada para permitir que un usuario registre múltiples vehículos.

Tabla: services

La tabla **services** define el catálogo cerrado de servicios disponibles en la plataforma. Este enfoque evita solicitudes ambiguas y mantiene control sobre el alcance del proyecto.

Cada servicio puede activarse o desactivarse sin afectar el resto del sistema.

Tabla: assistance_requests

Esta es la **tabla central del sistema**. Cada registro representa una solicitud real de asistencia vial. Todas las demás tablas se relacionan directa o indirectamente con esta entidad.

Aquí se almacena quién solicitó el servicio, qué tipo de servicio fue, dónde ocurrió, quién lo atendió y cuál fue su estado.

Tabla: request_status_logs

Los cambios de estado de una solicitud son críticos. Esta tabla permite registrar cada transición, quién la realizó y cuándo ocurrió.

Esto proporciona trazabilidad completa, soporte para auditorías y resolución de conflictos.

Tabla: quotes

Antes de cobrar, es importante calcular y registrar una cotización. Esta tabla permite almacenar el desglose del costo de un servicio, separando tarifas, comisiones y totales.

Esto mejora la transparencia y prepara al sistema para pagos reales.

Tabla: pricing_rules

Las tarifas no deben estar codificadas de forma fija. Esta tabla permite definir reglas de precios por zona y tipo de servicio, facilitando ajustes futuros sin cambios en el código.

Tabla: payments

La tabla **payments** registra el pago asociado a cada solicitud. Aunque en etapas iniciales pueda usarse en modo prueba, su estructura permite una transición sencilla a pagos reales.

Tabla: payment_transactions

Los pagos reales implican múltiples eventos (autorización, captura, reembolso). Esta tabla permite registrar cada transacción asociada a un pago, proporcionando un historial detallado.

Tabla: refunds

En situaciones de cancelación o error, es necesario manejar reembolsos. Esta tabla permite registrar dichos eventos de forma controlada y transparente.

Tabla: provider_payouts

El proveedor debe recibir su parte del pago. Esta tabla permite separar el cobro al usuario del pago al proveedor, facilitando el control financiero y la contabilidad básica.

Tabla: ratings

La calidad del servicio es clave. Esta tabla permite que los conductores evalúen el servicio recibido, generando retroalimentación y fomentando la mejora continua.

Tabla: messages

La comunicación entre conductor y proveedor es necesaria, pero debe ser controlada. Esta tabla registra los mensajes intercambiados durante una solicitud, manteniendo un historial para soporte.

Tabla: message_templates

Para evitar mensajes inapropiados y agilizar la comunicación, esta tabla permite definir mensajes preconfigurados según el rol del usuario.

Tabla: device_tokens

Las notificaciones push son esenciales para una app móvil. Esta tabla permite registrar los dispositivos asociados a cada usuario y gestionar el envío de notificaciones.

Tabla: notifications

Esta tabla almacena el historial de notificaciones enviadas, permitiendo confirmar entregas y ofrecer al usuario un registro de eventos importantes.

Tabla: attachments

En algunos casos es necesario adjuntar evidencia, como fotografías o comprobantes. Esta tabla permite asociar archivos a solicitudes de forma flexible.

Tabla: support_tickets

Los problemas operativos requieren atención formal. Esta tabla permite gestionar tickets de soporte relacionados con una solicitud, incluyendo su seguimiento y resolución.

Tabla: audit_logs

Finalmente, la tabla **audit_logs** registra acciones críticas realizadas por usuarios administrativos. Esto es fundamental para seguridad, control interno y análisis posterior.


dbdiagram.io

Script SQL

CREATE DATABASE zyga

CHARACTER SET utf8mb4

COLLATE utf8mb4_unicode_ci;

USE zyga;

Usuarios y consentimientos

```
CREATE TABLE users (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(150) NOT NULL,
  email VARCHAR(150) NOT NULL UNIQUE,
  telefono VARCHAR(30),
  password VARCHAR(255) NOT NULL,
  rol ENUM('conductor','proveedor','admin','operador') NOT NULL,
  estatus ENUM('activo','inactivo','bloqueado') DEFAULT 'activo',
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL
);

CREATE TABLE user_consents (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  user_id BIGINT UNSIGNED NOT NULL,
  tipo ENUM('terms','privacy','marketing') NOT NULL,
  version VARCHAR(50) NOT NULL,
  accepted_at DATETIME NOT NULL,
  ip VARCHAR(45),
  user_agent TEXT,
  FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Zonas de servicio

```
CREATE TABLE service_areas (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  estado VARCHAR(100),
  pais VARCHAR(100) DEFAULT 'México',
  activo BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL
);
CREATE TABLE area_polygons (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
```

```
  service_area_id BIGINT UNSIGNED NOT NULL,
  nombre VARCHAR(100),
  polygon_text LONGTEXT,
  FOREIGN KEY (service_area_id) REFERENCES service_areas(id)
);
```

Proveedores

```
CREATE TABLE providers (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  user_id BIGINT UNSIGNED NOT NULL UNIQUE,
  nombre_comercial VARCHAR(150),
  tipo_proveedor VARCHAR(100),
  validado BOOLEAN DEFAULT FALSE,
  estatus ENUM('activo','pendiente','suspendido') DEFAULT 'pendiente',
  service_area_id BIGINT UNSIGNED,
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL,
  FOREIGN KEY (user_id) REFERENCES users(id),
  FOREIGN KEY (service_area_id) REFERENCES service_areas(id)
);
CREATE TABLE provider_services (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  provider_id BIGINT UNSIGNED NOT NULL,
  service_id BIGINT UNSIGNED NOT NULL
);
CREATE TABLE provider_schedules (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  provider_id BIGINT UNSIGNED NOT NULL,
  day_of_week TINYINT NOT NULL,
  start_time TIME NOT NULL,
  end_time TIME NOT NULL,
  activo BOOLEAN DEFAULT TRUE,
  FOREIGN KEY (provider_id) REFERENCES providers(id)
);
CREATE TABLE provider_locations (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  provider_id BIGINT UNSIGNED NOT NULL,
  latitud DECIMAL(10,7),
  longitud DECIMAL(10,7),
  recorded_at DATETIME NOT NULL,
  FOREIGN KEY (provider_id) REFERENCES providers(id)
);
CREATE TABLE provider_documents (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  provider_id BIGINT UNSIGNED NOT NULL,
  tipo VARCHAR(50),
  file_url TEXT,
  estatus ENUM('pendiente','aprobado','rechazado') DEFAULT 'pendiente',
  expires_at DATE,
  reviewed_by BIGINT UNSIGNED,
  reviewed_at DATETIME,
  notes TEXT,
  FOREIGN KEY (provider_id) REFERENCES providers(id),
  FOREIGN KEY (reviewed_by) REFERENCES users(id)
```

```
);
```

Vehículos

```
CREATE TABLE vehicles (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  user_id BIGINT UNSIGNED NOT NULL,
  tipo ENUM('auto','moto') NOT NULL,
  marca VARCHAR(100),
  modelo VARCHAR(100),
  color VARCHAR(50),
  placas VARCHAR(20),
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL,
  FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Solicitudes de asistencia

```
CREATE TABLE assistance_requests (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  folio VARCHAR(50) NOT NULL UNIQUE,
  user_id BIGINT UNSIGNED NOT NULL,
  provider_id BIGINT UNSIGNED,
  service_id BIGINT UNSIGNED NOT NULL,
  vehicle_id BIGINT UNSIGNED,
  service_area_id BIGINT UNSIGNED NOT NULL,
  latitud DECIMAL(10,7),
  longitud DECIMAL(10,7),
  direccion_referencia VARCHAR(255),
  estatus
ENUM('creada','buscando','asignada','en_camino','en_proceso','finalizada'
,'cancelada') DEFAULT 'creada',
  cancel_reason VARCHAR(255),
  cancelled_by BIGINT UNSIGNED,
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL,
  FOREIGN KEY (user_id) REFERENCES users(id),
  FOREIGN KEY (provider_id) REFERENCES providers(id),
  FOREIGN KEY (service_id) REFERENCES services(id),
  FOREIGN KEY (vehicle_id) REFERENCES vehicles(id),
  FOREIGN KEY (service_area_id) REFERENCES service_areas(id),
  FOREIGN KEY (cancelled_by) REFERENCES users(id)
);
```

Tarifas y cotizaciones

```
CREATE TABLE pricing_rules (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
```

```
  service_area_id BIGINT UNSIGNED NOT NULL,
  service_id BIGINT UNSIGNED NOT NULL,
  nombre VARCHAR(100),
  base_fee DECIMAL(10,2),
  per_km_fee DECIMAL(10,2),
  night_surcharge DECIMAL(10,2),
  activo BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL,
  FOREIGN KEY (service_area_id) REFERENCES service_areas(id),
  FOREIGN KEY (service_id) REFERENCES services(id)
);

CREATE TABLE quotes (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  assistance_request_id BIGINT UNSIGNED NOT NULL UNIQUE,
  moneda VARCHAR(10) DEFAULT 'MXN',
  subtotal DECIMAL(10,2),
  comision_plataforma DECIMAL(10,2),
  total DECIMAL(10,2),
  pricing_rule_id BIGINT UNSIGNED,
  created_at TIMESTAMP NULL,
  FOREIGN KEY (assistance_request_id) REFERENCES assistance_requests(id),
  FOREIGN KEY (pricing_rule_id) REFERENCES pricing_rules(id)
);
```

Pagos

```
CREATE TABLE payments (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  assistance_request_id BIGINT UNSIGNED NOT NULL UNIQUE,
  monto DECIMAL(10,2),
  metodo ENUM('efectivo','tarjeta','digital'),
  estado_pago ENUM('pendiente','pagado','fallido','reembolsado'),
  provider_gateway VARCHAR(50),
  gateway_reference VARCHAR(150),
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL,
  FOREIGN KEY (assistance_request_id) REFERENCES assistance_requests(id)
);
CREATE TABLE payment_transactions (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  payment_id BIGINT UNSIGNED NOT NULL,
  tipo ENUM('authorization','capture','charge','refund'),
  estado ENUM('pending','succeeded','failed'),
  gateway_reference VARCHAR(150),
  raw_response JSON,
  created_at TIMESTAMP NULL,
  FOREIGN KEY (payment_id) REFERENCES payments(id)
);
CREATE TABLE refunds (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  payment_id BIGINT UNSIGNED NOT NULL,
  monto DECIMAL(10,2),
```

```
  motivo VARCHAR(255),
  estado ENUM('pending','succeeded','failed'),
  gateway_reference VARCHAR(150),
  created_at TIMESTAMP NULL,
  FOREIGN KEY (payment_id) REFERENCES payments(id)
);
CREATE TABLE provider_payouts (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  provider_id BIGINT UNSIGNED NOT NULL,
  assistance_request_id BIGINT UNSIGNED NOT NULL,
  monto DECIMAL(10,2),
  estado ENUM('pendiente','pagado'),
  payout_reference VARCHAR(150),
  created_at TIMESTAMP NULL,
  FOREIGN KEY (provider_id) REFERENCES providers(id),
  FOREIGN KEY (assistance_request_id) REFERENCES assistance_requests(id)
);
```

Evidencias, soporte y auditoría

```
CREATE TABLE attachments (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  assistance_request_id BIGINT UNSIGNED NOT NULL,
  uploaded_by BIGINT UNSIGNED NOT NULL,
  tipo VARCHAR(50),
  file_url TEXT,
  created_at TIMESTAMP NULL,
  FOREIGN KEY (assistance_request_id) REFERENCES assistance_requests(id),
  FOREIGN KEY (uploaded_by) REFERENCES users(id)
);
CREATE TABLE support_tickets (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  assistance_request_id BIGINT UNSIGNED NOT NULL,
  opened_by BIGINT UNSIGNED NOT NULL,
  assigned_to BIGINT UNSIGNED,
  categoria VARCHAR(50),
  prioridad ENUM('baja','media','alta'),
  estatus ENUM('abierto','en_proceso','resuelto','cerrado') DEFAULT
'abierto',
  descripcion TEXT,
  resolution TEXT,
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL,
  FOREIGN KEY (assistance_request_id) REFERENCES assistance_requests(id),
  FOREIGN KEY (opened_by) REFERENCES users(id),
  FOREIGN KEY (assigned_to) REFERENCES users(id)
);
CREATE TABLE audit_logs (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  actor_user_id BIGINT UNSIGNED NOT NULL,
  accion VARCHAR(150),
  entidad VARCHAR(100),
  entidad_id BIGINT UNSIGNED,
  metadata JSON,
  created_at TIMESTAMP NULL,
```

```
  FOREIGN KEY (actor_user_id) REFERENCES users(id)
);
```