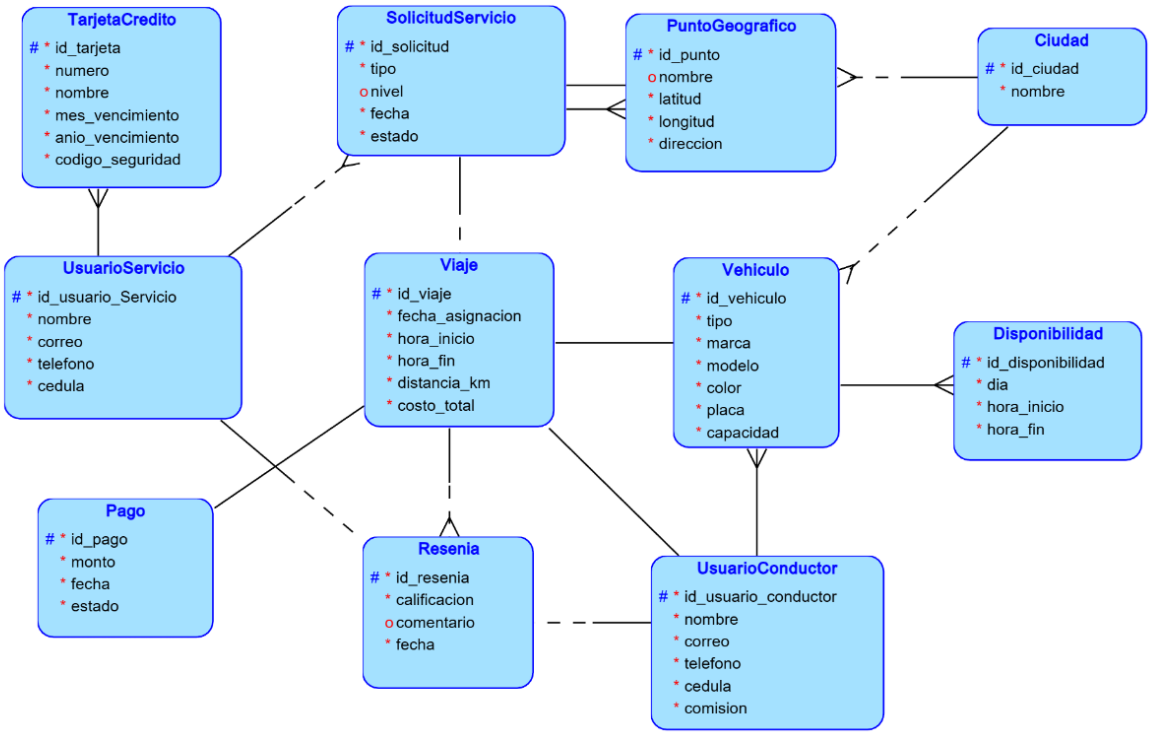
Entrega 3 – Proyecto Sistemas Transaccionales

Integrantes:

- Nicolás Hernández - 202322148
- Alejandro Cruz -
- Mariana Cediel - 202321548

1. Diseño de la base de datos

a) Modelo E/R



b) Lista de entidades con su descripción, y relaciones entre entidades con su cardinalidad

Entidad	Descripción
TarjetaCredito	Gestiona los datos de las tarjetas registradas por los usuarios (número, nombre, vencimiento, código de seguridad).
UsuarioServicio	Usuario que solicita viajes (cliente). Contiene su información personal y datos de contacto.

SolicitudServicio	Representa la solicitud que un usuario hace para pedir un viaje (tipo, nivel, fecha, estado).
PuntoGeografico	Lugares geográficos usados como origen/destino (nombre, latitud, longitud, dirección).
Ciudad	Registra las ciudades donde opera la aplicación.
Viaje	Contiene la información del viaje asignado (hora inicio/fin, distancia, costo total).
Vehiculo	Información del vehículo del conductor (tipo, marca, placa, color, capacidad).
Disponibilidad	Horario en el que un vehículo/conductor está disponible.
UsuarioConductor	Conductores registrados que prestan el servicio (datos personales y comisión).
Pago	Información de los pagos realizados por los usuarios (monto, fecha, estado).
Reseña	Opiniones/calificaciones que un usuario deja luego de un viaje.

Relación	Cardinalidad	Descripción
TarjetaCredito – UsuarioServicio	1:N	Un usuario puede registrar varias tarjetas; cada tarjeta pertenece a un solo usuario.
UsuarioServicio – SolicitudServicio	1:N	Un usuario puede hacer múltiples solicitudes de servicio.
SolicitudServicio – PuntoGeografico (origen/destino)	N:1 (origen) y N:1 (destino)	Cada solicitud tiene un origen y destino, pero un punto geográfico puede ser usado por muchas solicitudes.
PuntoGeografico – Ciudad	N:1	Un punto geográfico pertenece a una ciudad; muchas ubicaciones pueden estar en la misma ciudad.
SolicitudServicio – Viaje	1:1	Cada solicitud genera un solo viaje, y cada viaje proviene de una única solicitud.
Viaje – UsuarioServicio	N:1	Un usuario puede tener muchos viajes; cada viaje pertenece a un usuario.
Viaje – UsuarioConductor	N:1	Un conductor puede completar muchos viajes; cada viaje solo tiene un conductor asignado.
Viaje – Vehiculo	N:1	Un vehículo puede ser usado en muchos viajes; un viaje usa un solo vehículo.
Viaje – Pago	1:1	Cada viaje tiene un único pago asociado.

Vehiculo – Disponibilidad	1:N	Un vehículo tiene múltiples franjas de disponibilidad.
Viaje – Reseña	1:1	Cada viaje puede generar una única reseña.
UsuarioServicio – Reseña	1:N	Un usuario puede escribir muchas reseñas, pero cada reseña pertenece a un usuario.
UsuarioConductor – Vehiculo	1:N	Un conductor puede registrar varios vehículos; cada vehículo pertenece a un solo conductor.

c) Análisis de selección de esquema de asociación (modelo normalizado o embebido) para cada relación entre entidades, justificado a partir de los requerimientos de la aplicación.

Nuestra base de datos es una combinación de un modelo embebido con uno normalizado, con el fin de facilitar los requerimientos de consulta de la aplicación. El modelo tiene 3 colecciones independientes principales.

La primera es **Usuario_servicio**, que la decidimos poner como colección principal independiente ya que la consulta RFC1 requiere acceder rápidamente al historial de viajes de un usuario, y esto se logra almacenando los viajes como documentos separados que referencian al usuario. Además, el usuario puede tener múltiples tarjetas de crédito, las cuales no generan búsquedas pesadas ni relaciones externas, por lo que se decidió embeberlas dentro del usuario, optimizando lecturas sin aumentar la complejidad del sistema.

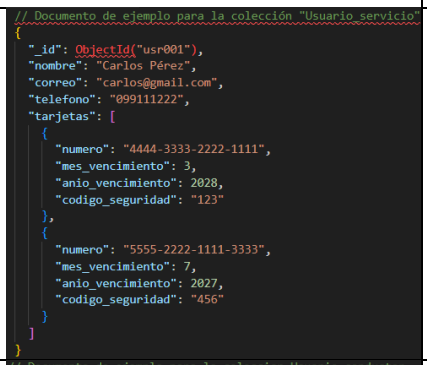
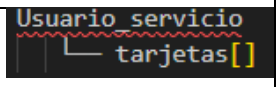
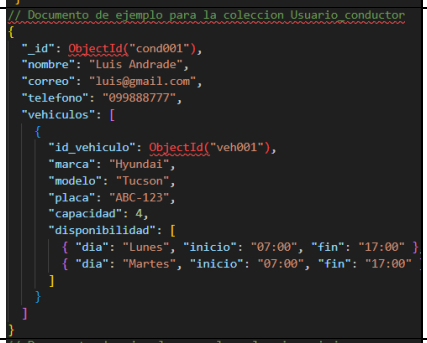
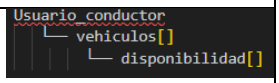
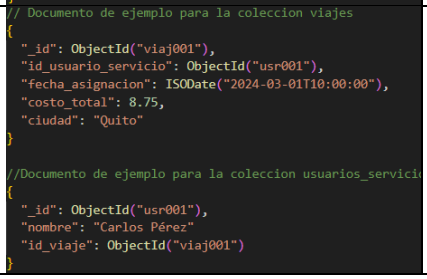
La segunda colección principal es **Usuario_conductor**, que también se mantiene independiente porque RFC2 demanda analizar qué conductores han realizado más viajes. Para esto es necesario que cada viaje referencie a un conductor, permitiendo agregaciones eficientes sin que el documento del conductor crezca indefinidamente. Sin embargo, información dependiente exclusivamente del conductor, como sus vehículos y sus horarios de disponibilidad, se embebe dentro de esta colección, ya que estos datos no requieren consultas transversales ni búsquedas por separado, y su tamaño es controlado, lo que mejora la coherencia y reduce la necesidad de múltiples colecciones.

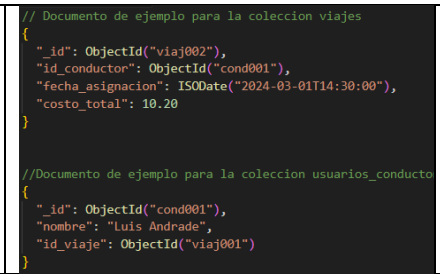
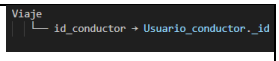
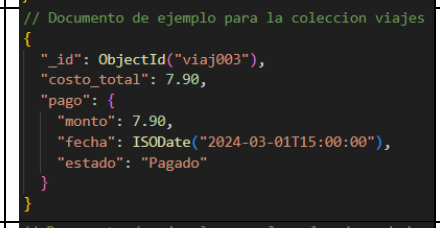
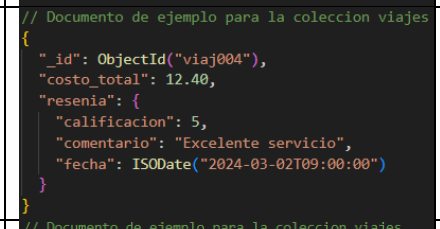
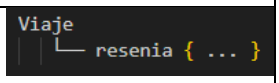
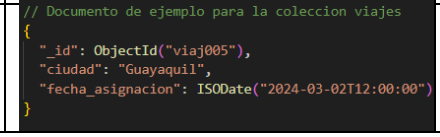
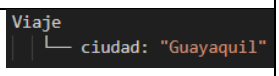
La tercera colección es **Viajes**, que fue diseñada como una colección normalizada y central, ya que todos los requerimientos operan sobre ella: RFC1 necesita filtrar por usuario, RFC2 agrupar por conductor y RFC3 filtrar por ciudad y rango de fechas. Por esta razón, los viajes almacenan referencias a Usuario_servicio y Usuario_conductor, pero al

mismo tiempo contienen algunos datos embebidos, como ciudad, pago y reseña. Estos elementos se mantienen dentro del documento porque están relacionados de manera uno a uno con el viaje y su acceso se realiza siempre desde este contexto, eliminando la necesidad de joins o lookups.

En conjunto, este modelo híbrido permite equilibrar dos objetivos: por un lado, evitar documentos demasiado grandes que perjudiquen la escalabilidad; y por otro, reducir la cantidad de operaciones de lectura mediante el uso de datos embebidos cuando estos dependen exclusivamente del documento principal. Con este diseño, las consultas solicitadas por los requerimientos funcionales de consulta se ejecutan de manera más eficiente.

d) Una descripción gráfica usando Json de cada relación entre entidades en donde presente un ejemplo de datos junto con el esquema de asociación usado (referenciado o embebido).

Relación	Tipo	Descripción gráfica usando JSON	Esquema de asociación
Usuario_servicio --> Tarjetas	Embebida 1:N		
Usuario_conductor--> Vehiculos	Embebida 1:N		
Viaje --> Usuario_servicio	Referenciada N:1		

Viaje --> Usuario_conductor	Referenciada N:1		
Viaje --> Pago	Embebida 1:1		
Viaje --> Resenia	Embebida 1:1		
Viaje --> Ciudad	Embebida 1:1		

2. Implementación de la base de datos

A continuación, se presentan los scripts utilizados para crear cada colección en nuestra base de datos.

>_MONGOSH

```
ISIS2304A14202520 > db.createCollection("usuario_conductor", {
                      validator: {$jsonSchema: {bsonType: "object",
                          required: ["nombre", "correo", "comision", "vehiculos"],
                          properties: {nombre: { bsonType: "string" },
                            correo: { bsonType: "string" },
                            comision: { bsonType: "double" },
                            vehiculos: {
                              bsonType: "array",
                              items: {
                                bsonType: "object",
                                required: [
                                  "tipo", "marca", "placa", "color", "capacidad", "disponibilidad"
                                ],
                                properties: {
                                  tipo: { bsonType: "string" },
                                  marca: { bsonType: "string" },
                                  placa: { bsonType: "string" },
                                  color: { bsonType: "string" },
                                  capacidad: { bsonType: "int" },
                                  disponibilidad: {
                                    bsonType: "array",
                                    items: {
                                      bsonType: "object",
                                      required: ["dia", "hora_inicio", "hora_fin"],
                                      properties: {
                                        dia: { bsonType: "string" },
                                        hora_inicio: { bsonType: "string" },
                                        hora_fin: { bsonType: "string" }
                                      }}}}}}}});
```

>_MONGOSH

```
ISIS2304A14202520> db.createCollection("usuario_conductor", {
                     validator: {
                       $jsonSchema: {
                         bsonType: "object",
                         required: ["nombre", "correo", "comision", "vehiculos"],
                         properties: { nombre: { bsonType: "string" },
                           correo: { bsonType: "string" },
                           comision: { bsonType: "double" },
                           vehiculos: { bsonType: "array",
                             items: {
                               bsonType: "object",
                               required: [
                                 "tipo", "marca", "placa", "color", "capacidad", "disponibilidad"
                               ],
                               properties: {
                                 tipo: { bsonType: "string" },
                                 marca: { bsonType: "string" },
                                 placa: { bsonType: "string" },
                                 color: { bsonType: "string" },
                                 capacidad: { bsonType: "int" },
                                 disponibilidad: {
                                   bsonType: "array",
                                   items: {
                                     bsonType: "object",
                                     required: ["dia", "hora_inicio", "hora_fin"],
                                     properties: {
                                       dia: { bsonType: "string" },
                                       hora_inicio: { bsonType: "string" },
                                       hora_fin: { bsonType: "string" }}}}}}}}});
```

```
>_MONGOSH
ISIS2304A14202520 > db.createCollection("viaje", {
                      validator: {
                        $jsonSchema: { bsonType: "object",
                          required: [
                            "usuario_servicio_id", "usuario_conductor_id",
                            "hora_inicio", "hora_fin", "distancia",
                            "costo_total", "ciudad", "pago"   ],
                          properties: {
                            usuario_servicio_id: { bsonType: "objectId" },
                            usuario_conductor_id: { bsonType: "objectId" },
                            hora_inicio: { bsonType: "date" },
                            hora_fin: { bsonType: "date" },
                            distancia: { bsonType: "double" },
                            costo_total: { bsonType: "double" },
                            ciudad: { bsonType: "object",
                              required: ["nombre"],
                              properties: {
                                nombre: { bsonType: "string" }} },
                            pago: { bsonType: "object",
                              required: ["monto", "fecha", "estado"],
                              properties: {
                                monto: { bsonType: "double" },
                                fecha: { bsonType: "date" },
                                estado: { bsonType: "string" } }  },
                            resena: { bsonType: ["object", "null"],
                              required: ["calificacion", "comentario"],
                              properties: {
                                calificacion: { bsonType: "int" },
                                comentario: { bsonType: "string" }}}}}});
```

```
ISIS2304A14202520 > db.createCollection("punto_geografico", {
                      validator: {
                        $jsonSchema: {
                          bsonType: "object",
                          required: ["nombre", "latitud", "longitud", "direccion", "ciudad_id"],
                          properties: {
                            nombre: { bsonType: "string" },
                            latitud: { bsonType: "double" },
                            longitud: { bsonType: "double" },
                            direccion: { bsonType: "string" },
                            ciudad_id: { bsonType: "objectId" }
                          }
                        }
                      }
                    });
```

```
ISIS2304A14202520 > db.createCollection("ciudad", {
                      validator: {
                        $jsonSchema: {
                          bsonType: "object",
                          required: ["nombre"],
                          properties: {
                            nombre: { bsonType: "string" }
                          }
                        }
                      }
                    });
```

```
ISIS2304A14202520> db.createCollection("solicitud_servicio", {
                     validator: {
                       $jsonSchema: {
                         bsonType: "object",
                         required: ["usuario_servicio_id", "tipo", "nivel", "fecha", "estado", "origen_id", "destino_id"],
                         properties: {
                           usuario_servicio_id: { bsonType: "objectId" },
                           tipo: { bsonType: "string" },
                           nivel: { bsonType: "string" },
                           fecha: { bsonType: "date" },
                           estado: { bsonType: "string" },
                           origen_id: { bsonType: "objectId" },
                           destino_id: { bsonType: "objectId" }
                         }
                       }
                     }
                   });
```

```
ISIS2304A14202520> db.createCollection("pago", {
                       validator: {
                         $jsonSchema: {
                           bsonType: "object",
                           required: ["monto", "fecha", "estado", "viaje_id"],
                           properties: {
                             monto: { bsonType: "double" },
                             fecha: { bsonType: "date" },
                             estado: { bsonType: "string" },
                             viaje_id: { bsonType: "objectId" }
                           }
                         }
                       }
                     });
```

3. Población de la base de datos

A continuación, se presentan imágenes de los scripts utilizados para poblar la base de datos.

```
ISIS2304A14202520> db.ciudad.insertMany([
                    {
                      _id: bogotaId,
                      nombre: "Bogotá"
                    },
                    {
                      _id: medellinId,
                      nombre: "Medellín"
                    }
                  ]);
```

```
ISIS2304A14202520> db.usuario_servicio.insertMany([
                  {
                    _id: usuario1Id,
                    nombre: "Carlos Pérez",
                    correo: "carlos.perez@mail.com",
                    telefono: "3001234567",
                    tarjetas: [  {
                        numero: "4111111111111111",
                        nombre: "Carlos Pérez",
                        vencimiento: "12/28",
                        codigo: "123"     }   ] },{
                    _id: usuario2Id,
                    nombre: "Ana Torres",
                    correo: "ana.torres@mail.com",
                    telefono: "3019876543",
                    tarjetas: [  {
                        numero: "5500000000000004",
                        nombre: "Ana Torres",
                        vencimiento: "07/26",
                        codigo: "456" },
                      {
                        numero: "340000000000009",
                        nombre: "Ana Torres",
                        vencimiento: "10/27",
                        codigo: "789"
                      }
                    ]
                  }
                ]);
```

```
ISIS2304A14202520> db.usuario_conductor.insertMany([
                    {
                      _id: conductor1Id,
                      nombre: "Luis Ramírez",
                      correo: "luis.ramirez@mail.com",
                      comision: 0.15,
                      vehiculos: [
                        {
                          tipo: "Automóvil",
                          marca: "Toyota",
                          placa: "ABC123",
                          color: "Rojo",
                          capacidad: 4,
                          disponibilidad: [
                            { dia: "Lunes", hora_inicio: "06:00", hora_fin: "18:00" },
                            { dia: "Martes", hora_inicio: "06:00", hora_fin: "18:00" }        ]      }    ]  },  {
                      _id: conductor2Id,
                      nombre: "María Gómez",
                      correo: "maria.gomez@mail.com",
                      comision: 0.20,
                      vehiculos: [     {
                          tipo: "Moto",
                          marca: "Honda",
                          placa: "XYZ987",
                          color: "Negro",
                          capacidad: 1,
                          disponibilidad: [
                            { dia: "Miércoles", hora_inicio: "07:00", hora_fin: "20:00" },
                            { dia: "Jueves", hora_inicio: "07:00", hora_fin: "20:00" }        ]      }    ]  }]);
```

```
ISIS2304A14202520> db.punto_geografico.insertMany([
                     {
                       _id: punto1Id,
                       nombre: "Centro Comercial Andino"
                       latitud: 4.6682,
                       longitud: -74.0534,
                       direccion: "Cra 11 #82-71",
                       ciudad_id: bogotaId
                     },
                     {
                       _id: punto2Id,
                       nombre: "Aeropuerto El Dorado",
                       latitud: 4.7016,
                       longitud: -74.1469,
                       direccion: "Av. Eldorado",
                       ciudad_id: bogotaId
                     },
                     {
                       _id: punto3Id,
                       nombre: "Parque Lleras",
                       latitud: 6.2088,
                       longitud: -75.5650,
                       direccion: "El Poblado",
                       ciudad_id: medellinId
                     }
                   ]);
```

```
ISIS2304A14202520> db.solicitud_servicio.insertMany([
                     {
                       _id: solicitud1Id,
                       usuario_servicio_id: usuario1Id,
                       tipo: "Económico",
                       nivel: "Básico",
                       fecha: new Date("2025-01-10T10:00:00Z"),
                       estado: "Completado",
                       origen_id: punto1Id,
                       destino_id: punto2Id
                     },
                     {
                       _id: solicitud2Id,
                       usuario_servicio_id: usuario2Id,
                       tipo: "Premium",
                       nivel: "Alto",
                       fecha: new Date("2025-01-12T15:30:00Z"),
                       estado: "Completado",
                       origen_id: punto3Id,
                       destino_id: punto1Id
                     }
                   ]);
```

```
ISIS2304A14202520> db.viaje.insertMany([{  _id: viaje1Id,
                      usuario_servicio_id: usuario1Id,
                      usuario_conductor_id: conductor1Id,
                      hora_inicio: new Date("2025-01-10T10:05:00Z"),
                      hora_fin: new Date("2025-01-10T10:40:00Z"),
                      distancia: 12.5,
                      costo_total: 26000,
                      ciudad: { nombre: "Bogotá" },
                      pago: {
                        monto: 26000,
                        fecha: new Date("2025-01-10T10:45:00Z"),
                        estado: "Pagado"    },
                      resena: {
                        calificacion: 5,
                        comentario: "Excelente servicio, muy amable."  }},{
                      _id: viaje2Id,
                      usuario_servicio_id: usuario2Id,
                      usuario_conductor_id: conductor2Id,
                      hora_inicio: new Date("2025-01-12T15:40:00Z"),
                      hora_fin: new Date("2025-01-12T16:20:00Z"),
                      distancia: 8.2,
                      costo_total: 18000,
                      ciudad: { nombre: "Medellín" },
                      pago: {
                        monto: 18000,
                        fecha: new Date("2025-01-12T16:25:00Z"),
                        estado: "Pagado"   },
                      resena: {
                        calificacion: 4, comentario: "Buen viaje, aunque un poco lento."}}]);
```

Requerimientos de consultas

RFC1
let usuarioId = ObjectId(“ID” que se busca)

db.solicitud_servicio.aggregate([

{ $match: { usuario_servicio_id: usuarioId } },

{ $lookup: { from: "usuario_servicio", localField: "usuario_servicio_id", foreignField: "_id", as: "usuario" } },

{ $unwind: "$usuario" },

{ $lookup: { from: "punto_geografico", localField: "origen_id", foreignField: "_id", as: "origen" } },

{ $unwind: "$origen" },

{ $lookup: { from: "punto_geografico", localField: "destino_id", foreignField: "_id", as: "destino" } },

{ $unwind: "$destino" },

{ $project: { _id: 0, fecha: 1, tipo: 1, nivel: 1, estado: 1, "usuario.nombre": 1, "usuario.correo": 1, "origen.nombre": 1, "origen.direccion": 1, "destino.nombre": 1, "destino.direccion": 1 } },

{ $sort: { fecha: -1 } }

])

RFC2

db.viaje.aggregate([

{ $group: { _id: "$usuario_conductor_id", numeroServicios: { $sum: 1 } } },

{ $sort: { numeroServicios: -1 } },

{ $limit: 20 },

{ $lookup: { from: "usuario_conductor", localField: "_id", foreignField: "_id", as: "conductor" } }, { $unwind: "$conductor" },

{ $project: { _id: 0, conductorId: "$_id", nombre: "$conductor.nombre", correo: "$conductor.correo", comision: "$conductor.comision", vehiculos: "$conductor.vehiculos", numeroServicios: 1 } }

]);

RFC3

let ciudadNombre = "Bogotá";

let fechaInicio = ISODate("2025-01-01T00:00:00Z");

let fechaFin   = ISODate("2025-12-31T23:59:59Z");

db.solicitud_servicio.aggregate([

```
{$match: {fecha: { $gte: fechaInicio, $lte: fechaFin }}}

{$lookup: {from: "punto_geografico", localField: "origen_id", foreignField: "_id", as: "origen"}},

{ $unwind: "$origen" },

{$lookup: {from: "ciudad", localField: "origen.ciudad_id", foreignField: "_id", as: "ciudad"}},

{ $unwind: "$ciudad" },

{$match: {"ciudad.nombre": ciudadNombre}},

{$group: {_id: { tipo: "$tipo", nivel: "$nivel" }, numeroServicios: { $sum: 1 }}},

{$group: {_id: null, totalServicios: { $sum: "$numeroServicios" }, detalles: {$push: {tipo: "$_id.tipo", nivel: "$_id.nivel", numeroServicios: "$numeroServicios"}}}},

{ $unwind: "$detalles" },

{$project: { _id: 0, tipo: "$detalles.tipo", nivel: "$detalles.nivel", numeroServicios: "$detalles.numeroServicios", porcentaje: {$multiply: [{$divide: ["$detalles.numeroServicios", "$totalServicios"] }, 100]}}},

{ $sort: { numeroServicios: -1 } }

]);
```