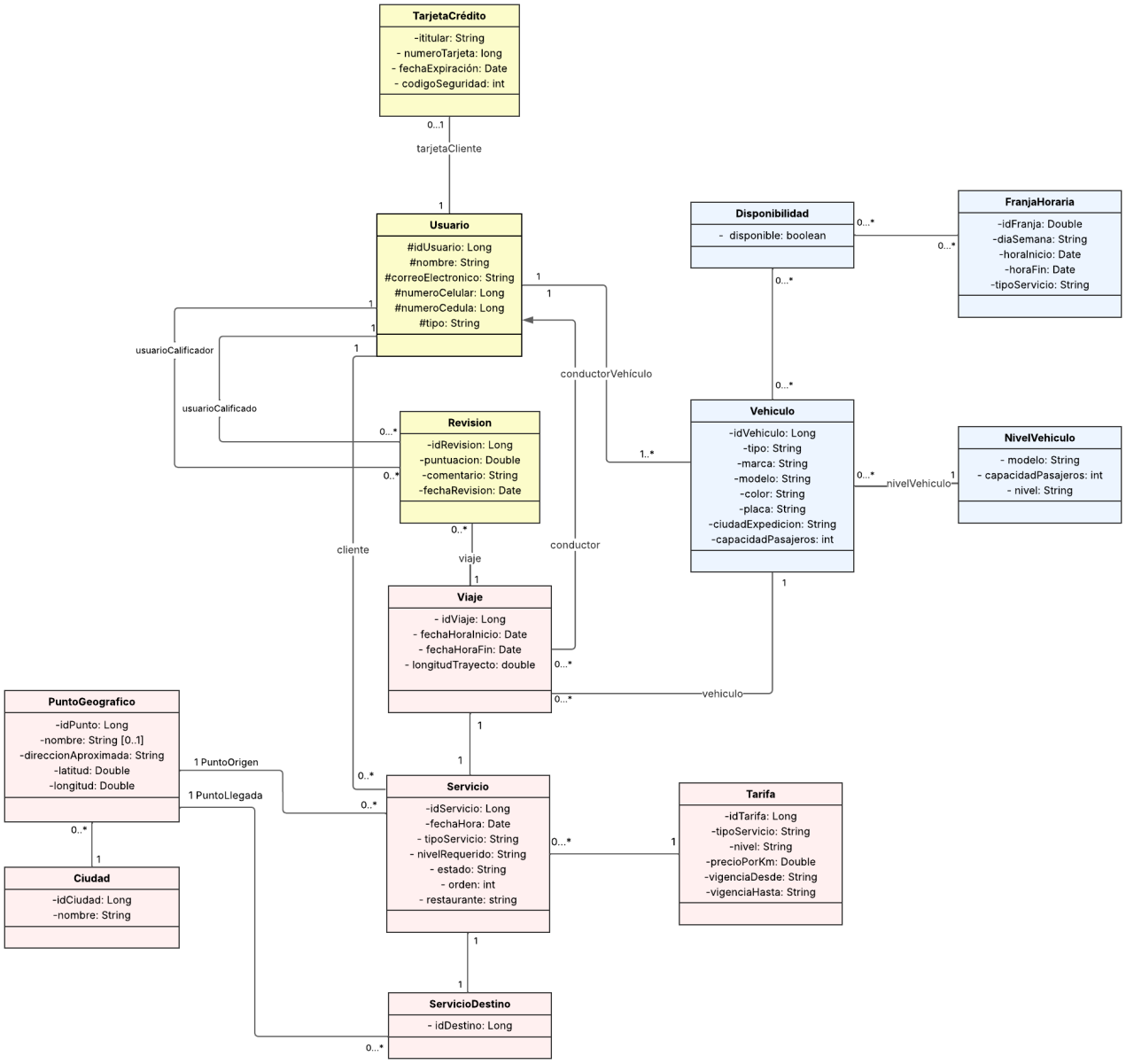
Entrega 3 AlpesCab

Sergio Arias | 202412370
Sara González | 202210908
Laura Martinez-Galindo | 202125643

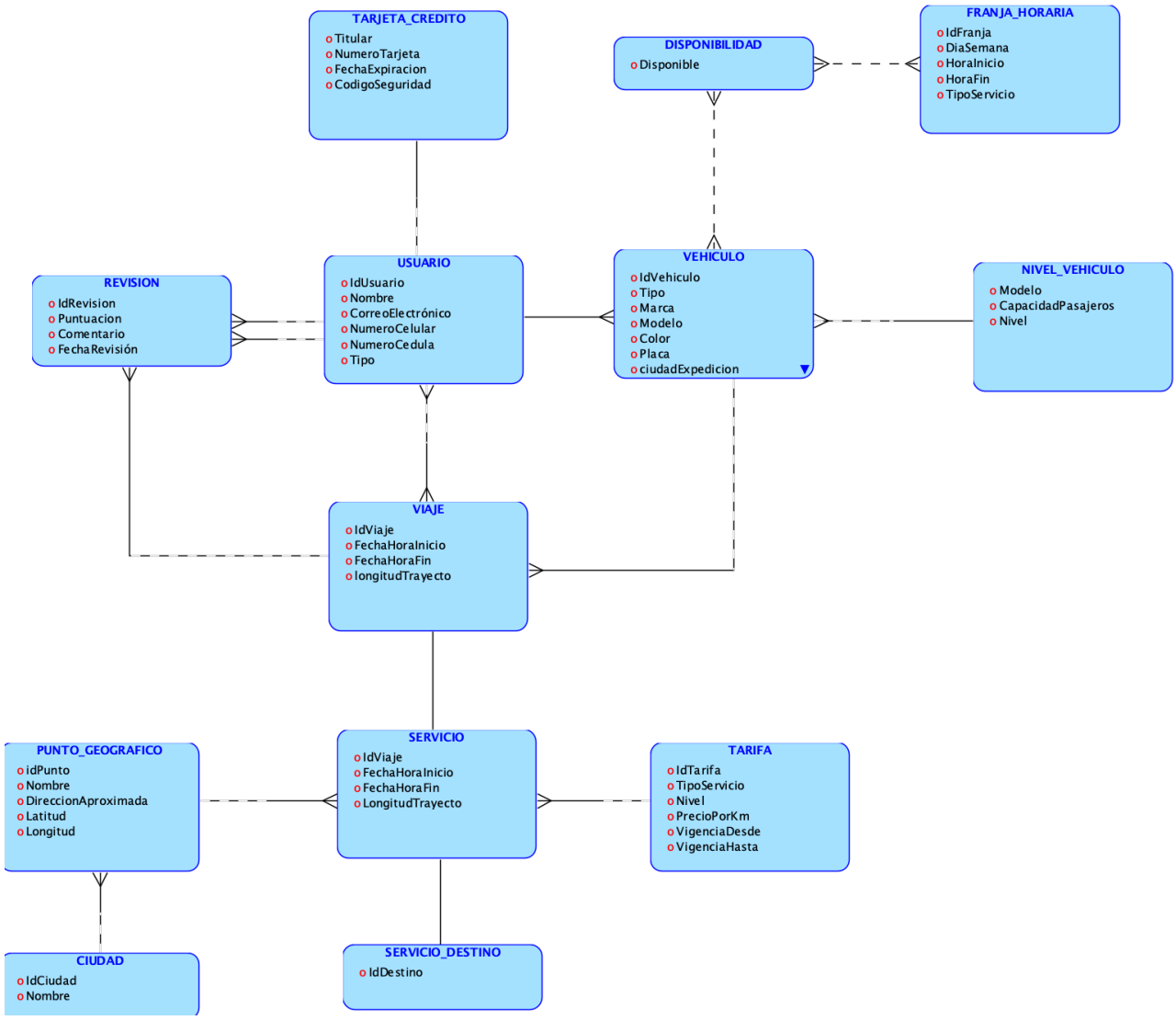
0. Modelamiento de AlpesCab

a. Modelo Conceptual UML



[Modelo en LucidChart para mejor visualización]

b. Modelo Conceptual E/R



c. Diseño de la base de datos y análisis de normalización

Se afirma que el modelo propuesto cumple con la Forma Normal de Boyce-Codd (BCNF), ya que se garantiza la eliminación de redundancias y anomalías de actualización. Cada entidad dentro del diagrama tiene una llave primaria bien definida, bien sea por el id o por una tupla de atributos, que determina de manera completa y única el resto de los atributos. Por ejemplo, en entidades como Vehículo, Usuario o FranjaHoraria, se observan los ID que se establecen por el sistema para facilitar la unicidad y el acceso a la información relevante; mientras que, en NivelVehiculo, por ejemplo, la llave primaria está compuesta por el modelo y la cantidad de pasajeros del automóvil y no hay otra llave candidata que comparta atributos con esta. Asimismo, no se presentan dependencias parciales ni transitivas, y las relaciones de muchos a muchos se encuentran correctamente descompuestas en tablas intermedias, como en los casos de *Revisión*, *ServicioDestino* o *Disponibilidad*. Adicional a ello, los atributos que podrían

generar jerarquías o multivariación, como los niveles de vehículo o las ciudades, se modelan como entidades independientes, evitando dependencias entre atributos no clave dentro de una misma tabla. Gracias a estas decisiones de diseño, existe integridad, consistencia y flexibilidad para futuras extensiones. También se realizaron simplificaciones y separaciones de tablas con el fin de que poder acceder y cumplir con los requerimientos funcionales junto con los de consulta, ejemplos de esto fueron las desapariciones de las tablas de usuarioServicio y usuarioConductor, quedando una única de Usuario.

Así luce el modelo relacional final, cuyas relaciones están en Forma Normal de Boyce-Codd (BCNF) como ya se justificó:

Usuario

idUsuario	nombre	numeroCelular	numeroCedula	correoElectronico	tipoUsuario
PK, SA	NN	NN	UA, NN, ND	UA, NN, ND	CK("Cliente", "Conductor")
1	Jorge	3004567890	1012456789	jaconductor@gamil.com	cliente
2	Tatiana	3109876543	1023456781	nierfan@gmail.com	conductor
3	Carlos	3206543871	1034567892	9sbstchr@gmail.com	cliente
4	Luka	3156781234	1045678912	LukaAnchor@gmail.com	cliente
5	Stinson	3155755237	1056789123	barney999@gmail.com	conductor
6	Robin	3112345678	1105912091	nightwing771@gmail.com	cliente

Review

idRevision	idUsuarioCalificador	idUsuarioCalificado	puntuacion	comentario	fecha	idViaje
PK,NN,SA,ND	FK(Usuario.idUsuario),NN,CK	FK(Usuario.idUsuario),NN,CK	NN, CK	SA, CK	NN	FK(Viaje.idViaje)
1	1	3	3,5	Va muy lento	8/31/2025	1
2	2	2	5	Tiene muy buena conversación	8/31/2025	2
3	3	2	4,8	Gran cliente	8/31/2025	3
4	2	1	3	Se salta los semaforos	8/31/2025	4
5	1	1	1	Iba viendo tiktok manejando	8/31/2025	5

ConductorVehiculo

idConductor	idVehiculo
PK, FK(Usuario.idUsuario),SA	PK, FK(Vehiculo.idVehiculo), NN, ND
4	1
4	2
5	3

TarjetaCredito

idTarjetaCredito	titularDeLaTarjeta	numeroTarjeta	fechaExpiracion	codigoSeguridad	clienteId
PK,NN,SA,ND,CK	CK, ND,UA, NN	NN, ND	NN	NN,CK	NN, FK(Usuario.idUsuario), CK(Usuario.tipo="Cliente")
1	Barney Stinson	123456789	27/11	245	1
2	MADRE DE TATIANA	987654321	27/01	111	2
3	Dick Grayson	123789456	32/02	222	3

Vehiculo

idVehiculo	tipo	marca	modelo	color	placa	ciudadExpedicion	capacidadPasajeros
NN,PK,SA,ND	NN	NN	NN	NN	NN,ND	NN	NN
1	Carro	Toyota	Colorado	Negro	ABC123	Bogotá	4
2	Motocicleta	Yamaha	Fast4X30	Rojo	DEF123	Cali	4
3	Camioneta	Nissan	Quest	Café	ABC456	Popayán	6
4	Carro	Dodge	Journey	Blanco	HIJ123	Chía	2
5	Motocicleta	Yamaha	Fast4X30	Negro	DEF456	Envigado	3

FranjaHoraria

idFranja	diaSemana	horaInicio	horaFin	tipoServicio
PK,SA,ND	NN	NN	NN	NN, CK(in "Transporte Pasajeros", "Domicilio Comida", "Transporte Paquete")
1	Lunes	9:30	11:30	Transporte Pasajeros
2	Lunes	14:00	19:00	Domicilio Comida
3	Jueves	6:00	13:00	Transporte Paquete
4	Viernes	20:00	24:00:00	Transporte Pasajeros
5	Domingo	7:00	14:00	Domicilio Comida

NivelVehiculo

modelo	capacidadPasajeros	nivel
PK, NN	PK, NN, CK(>0)	CK(in "Estandar", "Confort", "Large")
Colorado	4	Estandar
Fast4X30	4	Confort
Quest	6	Large
Journey	2	Confort
Fast4X30	3	Estandar

Disponibilidad

idVehiculo	idFranja	disponible
PK,FK(Vehiculo.idVehiculo)	PK, FK(FranjaHoraria.idFranja)	NN
1	1	true
1	2	true
1	3	false
4	1	true
5	5	true

PuntoGeografico

idPunto	nombre	latitud	longitud	direccionAproximada	idCiudad
PK,SA	NULL	NN	NN	NN	NN, FK(Ciudad.idCiudad)
1	Universidad de los Andes	4,61104	-74,070288	Carrera 7 con Calle 24	1
2	NULL	4,591053	-74,072167	ntre Carreras Séptima y Octava con Calles Décima	1
3	Parque Arví	4,65	-74,1	En la localidad de Teusaquillo, atravesado por las carrera	1
4	NULL	4,53604	-74,1388	Carrera 5 #12-41	1
5	NULL	0	0	Centro	5

Ciudad

idCiudad	nombre
PK	NN, ND
1	Bogotá
2	Cali
3	Medellín
4	Barranquilla
5	Santa Marta

Servicio

idServicio	idCliente	fechaHora	tipoServicio	nivelRequerido	estado	orden	restaurante	idPuntoPartida
PK, SA	FK(UsuarioServicios.idUsuario), NN	NN	NN, CK(in "Transporte Pasajeros", "Domicilio Comida", "Transporte Paquete")	CK(in "Estandar", "Confort", "Large"), NULL	NN, CK(in "Pendiente", "Asignado", "Cancelado")			FK(PuntoGeografico.idPunto), NN
1	2	23/09/2025 9:05	Transporte Pasajeros	Estandar	Pendiente	NULL	NULL	1
2	5	23/09/2025 9:05	Domicilio Comida	Confort	Asignado	34	El Corral	2
3	6	23/09/2025 9:05	Transporte Paquete	Large	Cancelado	NULL	NULL	3
4	2	23/09/2025 9:05	Transporte Pasajeros	Confort	Pendiente	NULL	NULL	4
5	5	23/09/2025 9:05	Domicilio Comida	Estandar	Pendiente	12	Crepes & Waffles	5

ServicioDestino

idDestino	idServicio	idPuntoLlegada
PK, SA	FK(Servicio.idServicio), NN	FK(PuntoGeografico.idPunto), NN
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

Tarifa

idTarifa	tipoServicio	nivel	precioPorKm	vigenciaDesde	vigenciaHasta
PK, NN, SA, ND	NN, CK(in "Transporte Pasajeros", "Domicilio Comida", "Transporte Paquete")	CK(in "Estandar", "Confort", "Large")	NN, CK(precioPorKm > 0)	NN, CK(vigenciaDesde<vigenciaHasta)	NN, CK(vigenciaHasta>vigenciaDesde)
1	Transporte Pasajeros	Estandar	2000	1/01/2025	12/31/2025
2	Domicilio Comida	Confort	2200	1/01/2025	12/31/2025
3	Transporte Paquete	Large	2500	1/01/2025	12/31/2025
4	Transporte Pasajeros	Confort	2800	1/01/2025	12/31/2025
5	Domicilio Comida	Estandar	1800	1/01/2025	12/31/2025

Viaje (este es el historico)

idViaje	fechaHoraInicio	fechaHoraFin	longitudTrayecto	idServicio	idConductor	idVehiculo
PK, NN, SA, ND	NN	NN	NN	FK(Servicio.idServicio), NN	FK(UsuarioConductor.idUsuario), NN	FK(Vehiculo.idVehiculo), NN
1	8:15:00	8:45:00	12.5	1	1	1
2	9:00:00	9:35:00	8.2	2	3	2
3	10:10:00	10:45:00	5.6	3	4	3
4	11:20:00	11:55:00	9.0	4	3	3
5	12:30:00	13:10:00	15.0	5	4	4

De igual forma se adjunta en los documentos del repositorio.

d. Instrucciones Base de Datos

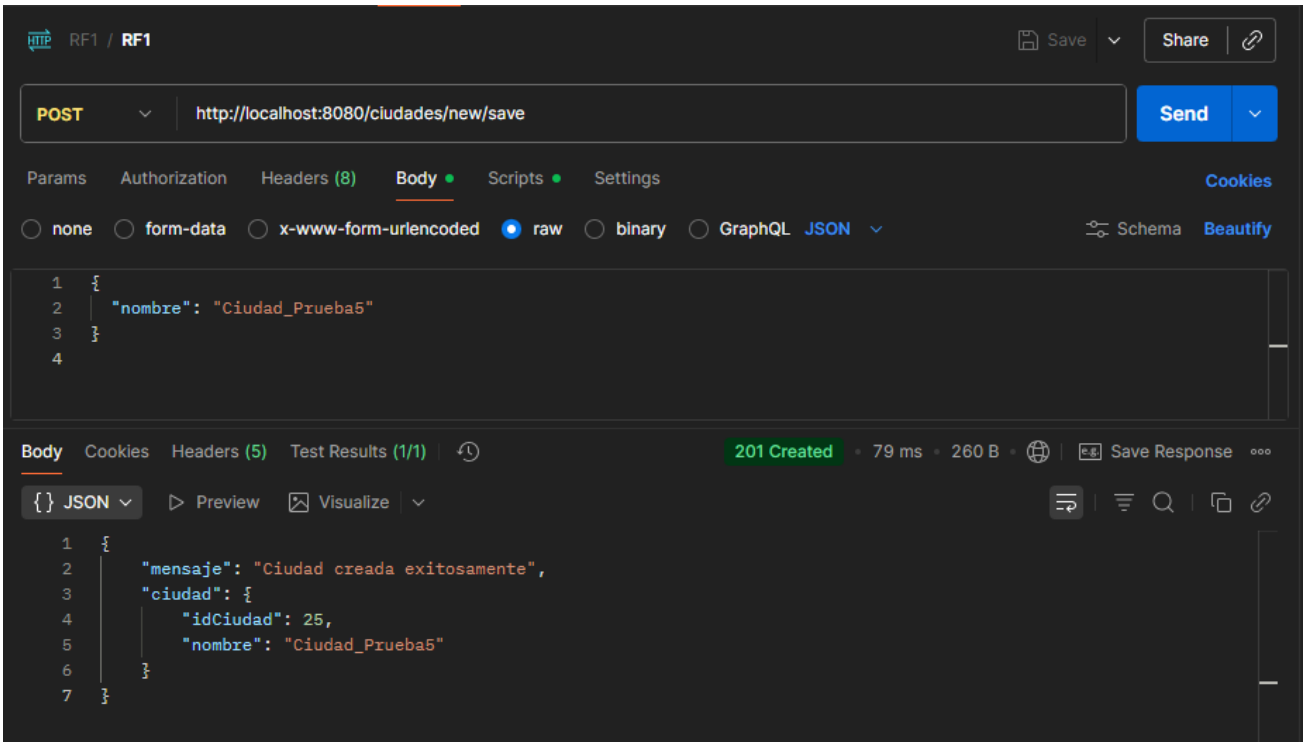
El primer paso es crear las tablas y las secuencias. En la ruta “src\db\nuevasTablas.sql”, se encuentra el script para generar las tablas del modelo. De igual forma, se generaron secuencias para los ids de algunas tablas, véase esto en “src\db\crearSecuencias.sql”.

Posteriormente, se deben poblar las tablas, para lo cual se dispone del archivo en “src\db\poblarTablas.sql”. La base de datos con las tablas, secuencias y los datos ya insertados se encuentra en el usuario ISIS2304A01202520 con contraseña kuuuULHdgiAZ.

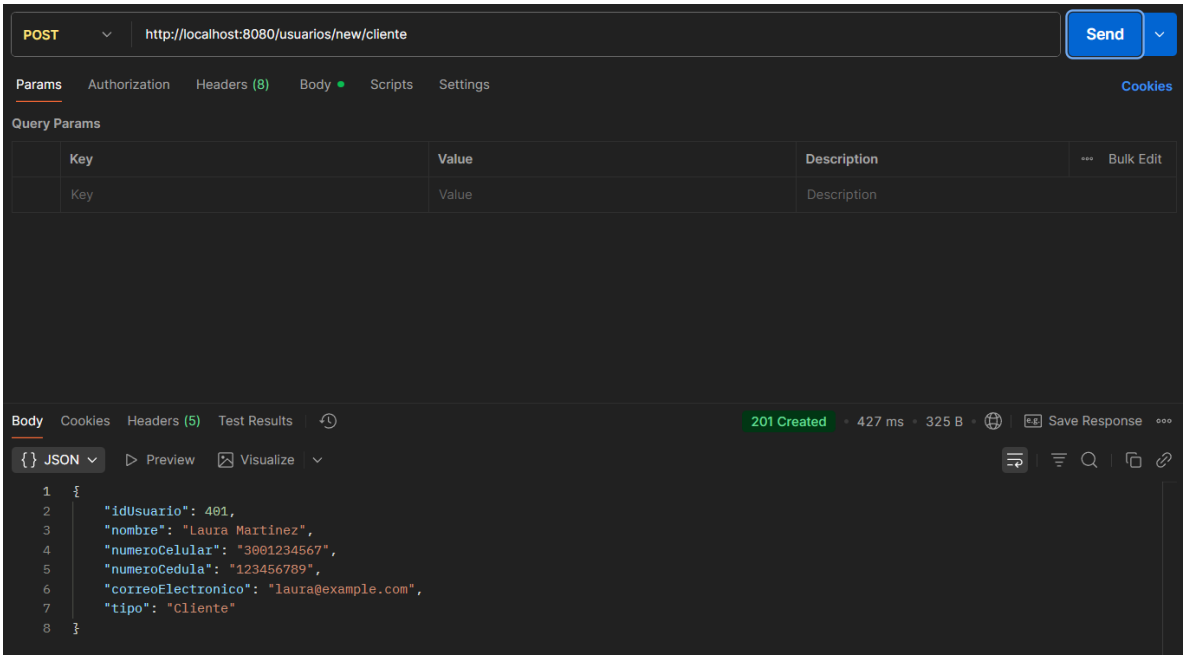
1. Implementación de la Aplicación Transaccional

Siguiendo el framework Spring, se realizaron las entidades, repositorios, controladores y servicios de cada una de las tablas o relaciones de nuestro modelo, todo se encuentra en el repositorio. Adicionalmente, se implementó tanto la lógica como la transaccionalidad para los requerimientos funcionales y de consulta, y se diseñaron escenarios de prueba en Postman para probar cada uno de ellos. A continuación, se presentan los resultados de estos escenarios:

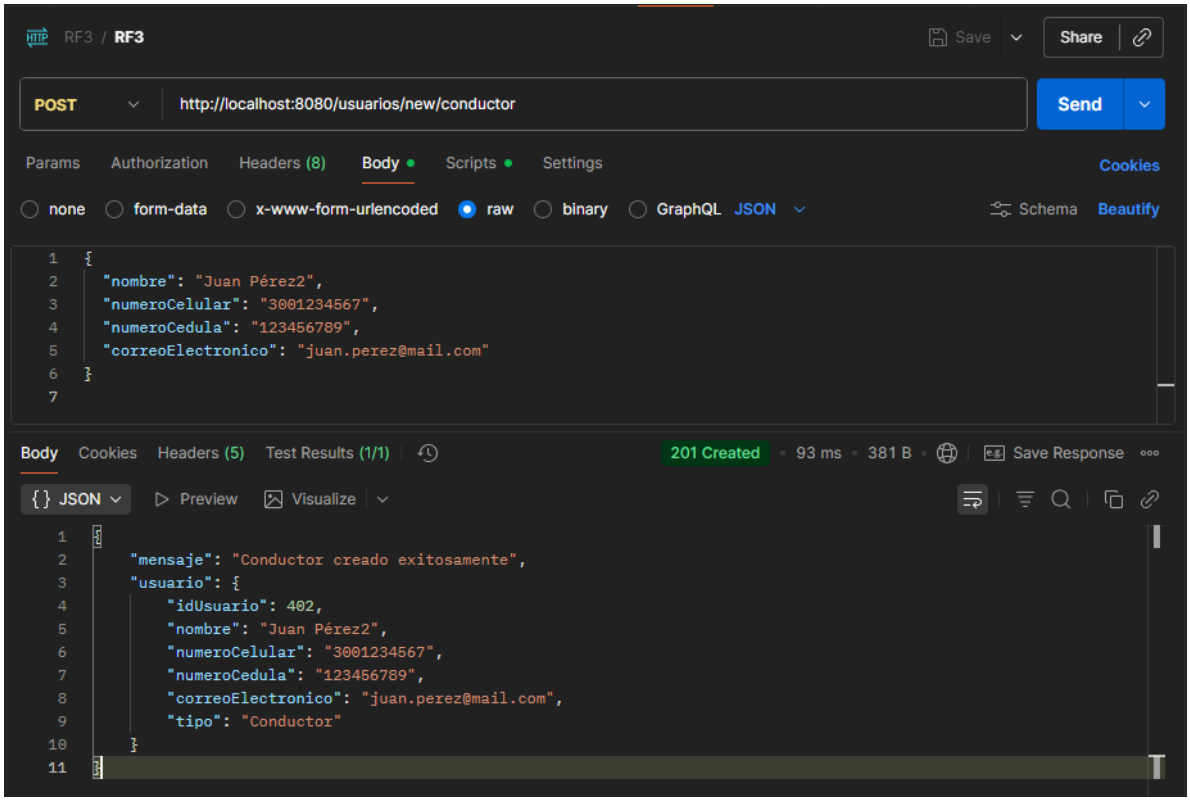
RF1: Registrar una ciudad



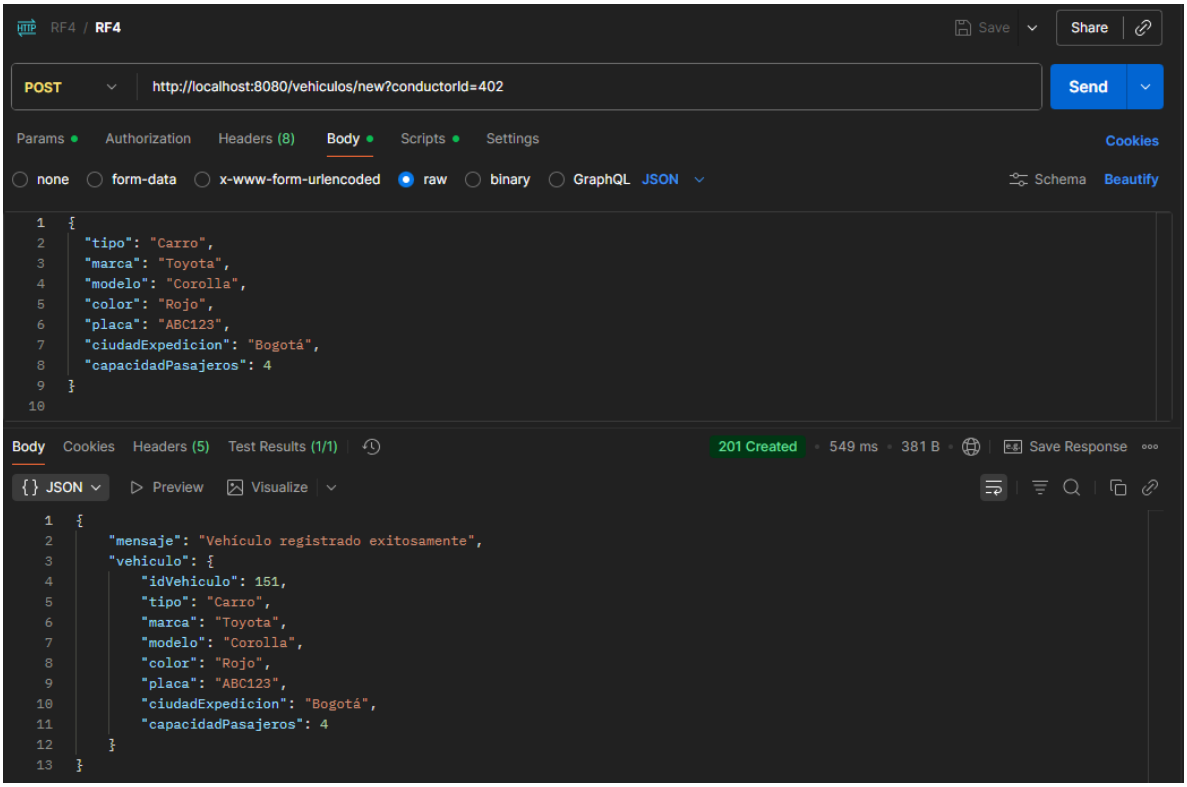
RF2: Registrar un usuario de servicios



RF3: Registrar un usuario conductor



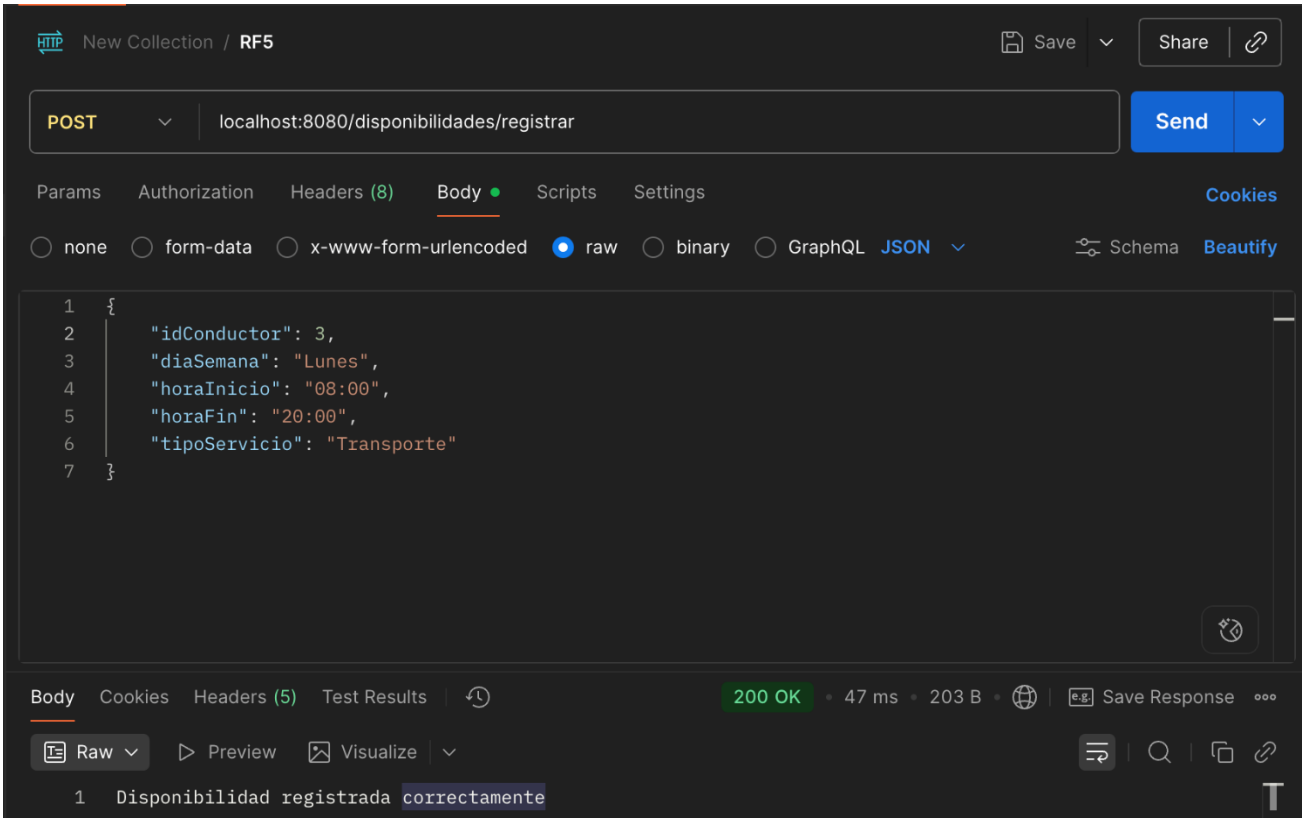
RF4: Registrar un vehículo para un usuario conductor



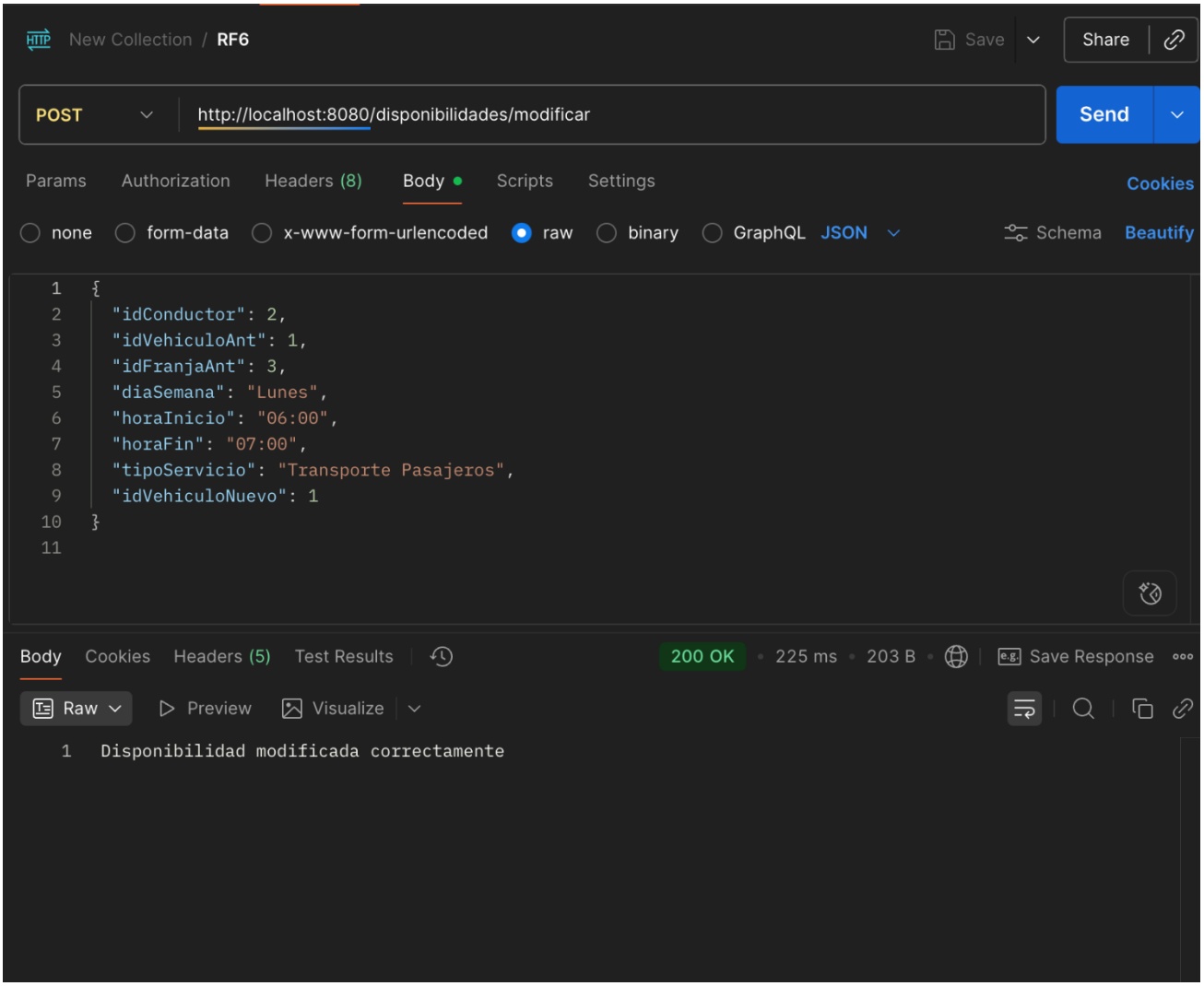
151	151	Carro	Toyota	Corolla	Rojo	ABC123	Bogotá

151	402	151

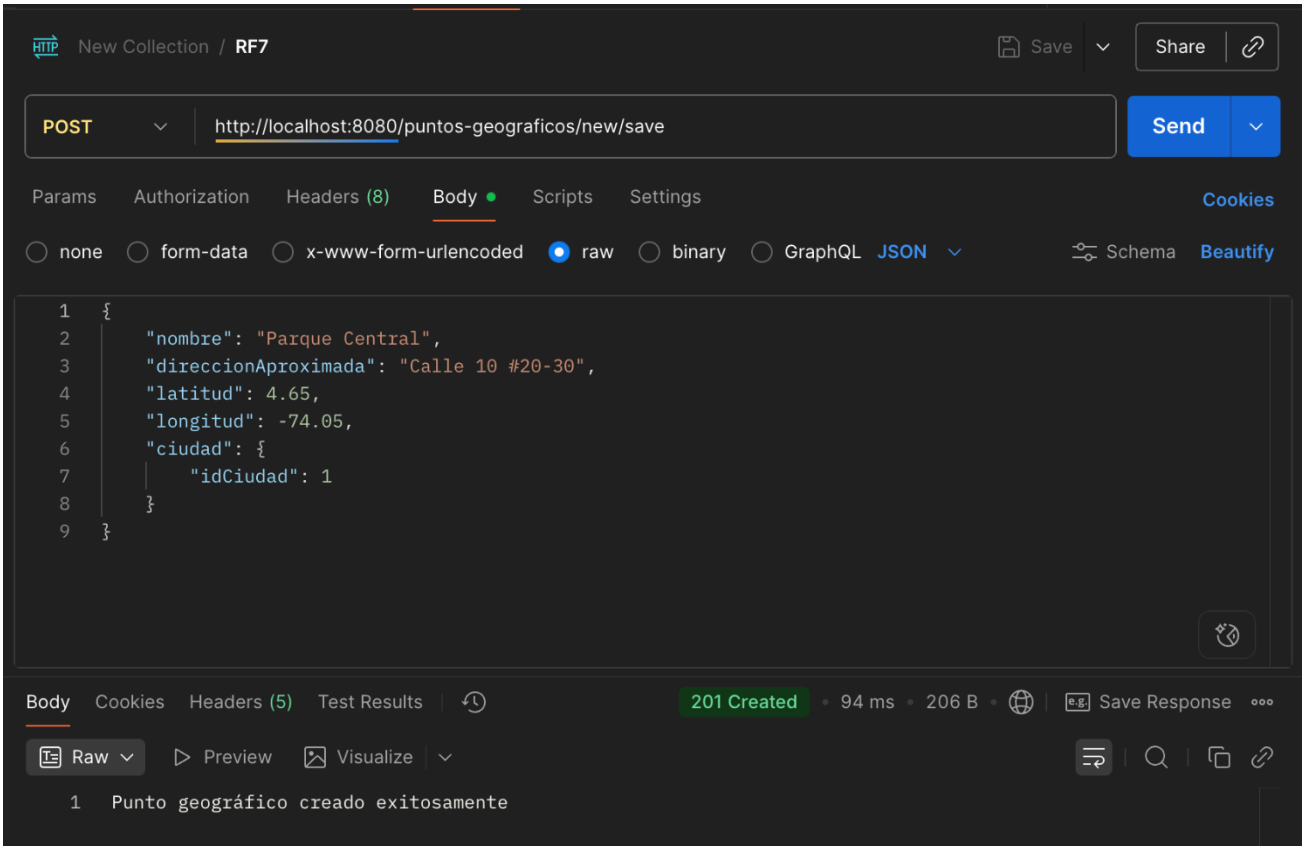
RF5: Registrar la disponibilidad de un vehículo para servicios



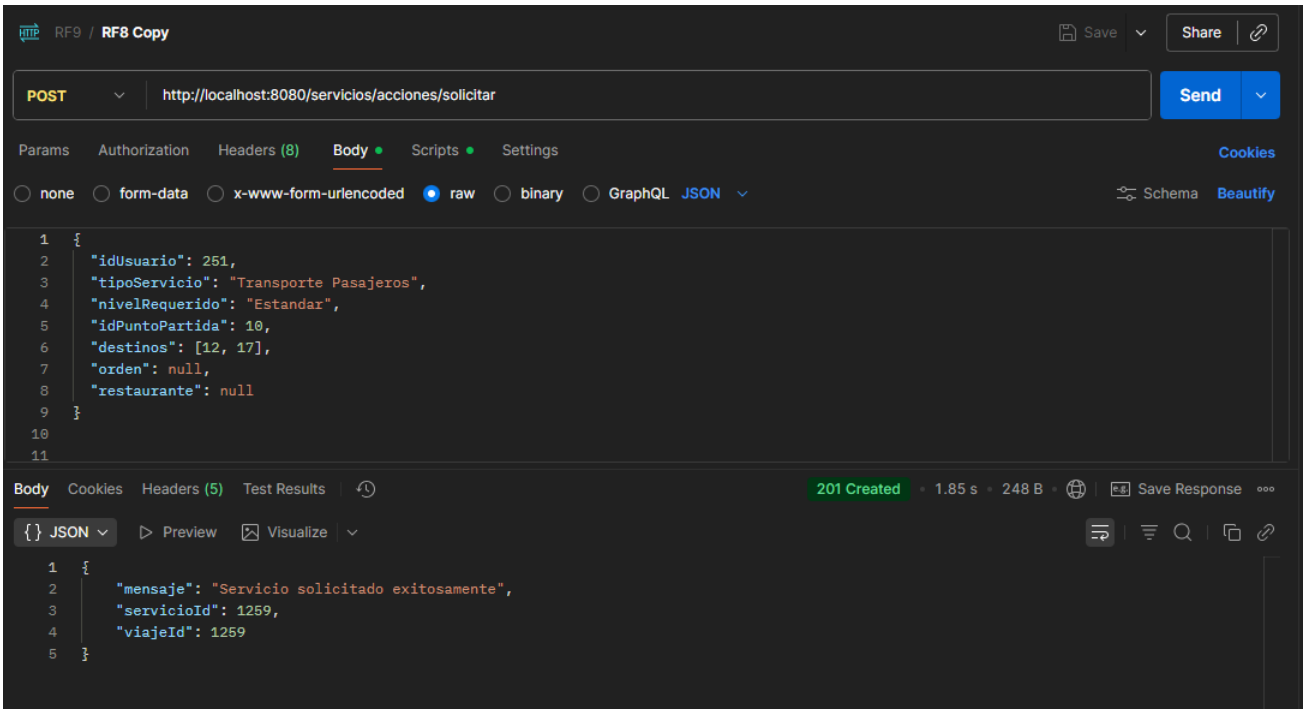
RF6: Modificar la disponibilidad de un vehículo para servicios



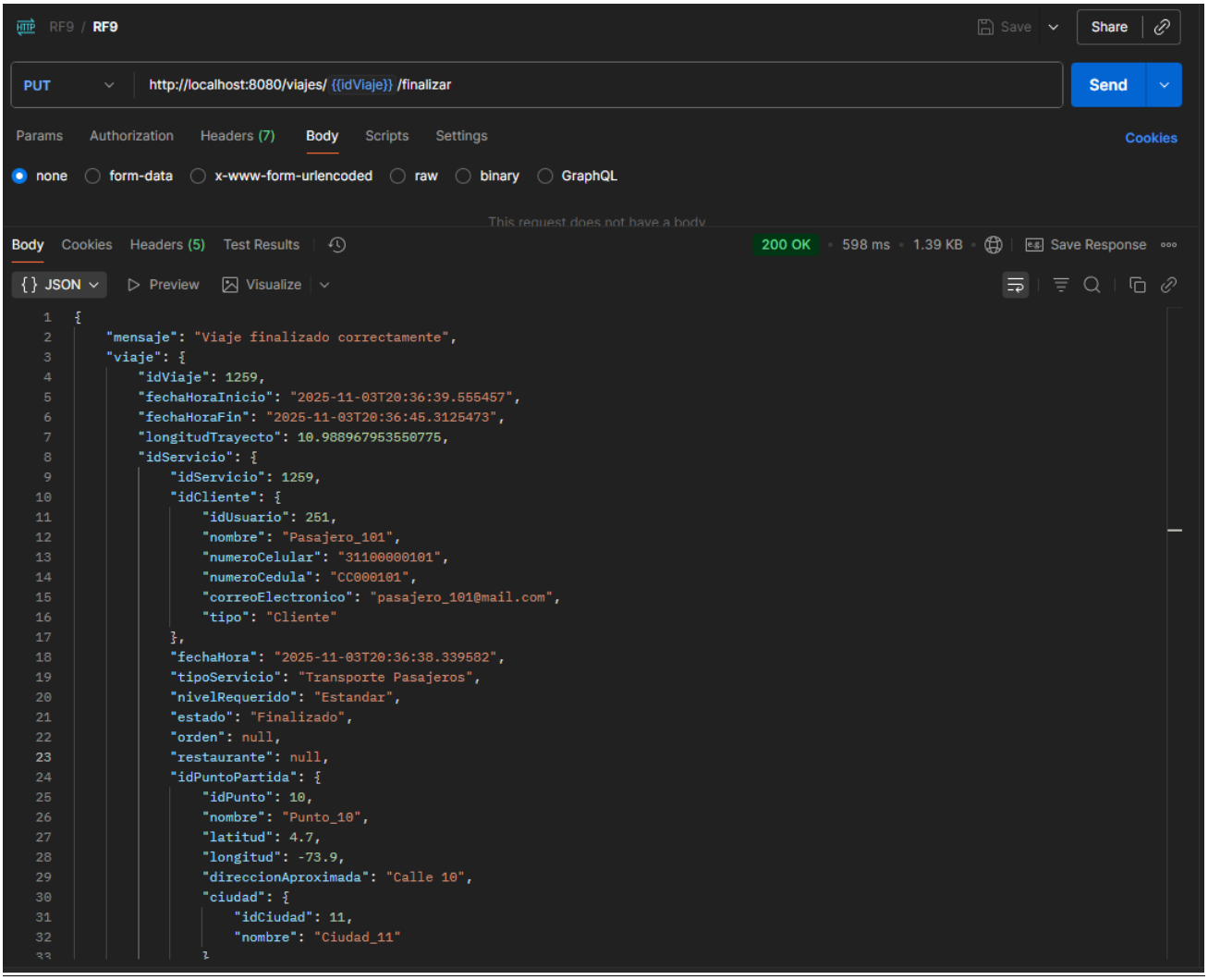
RF7: Registrar un punto geográfico

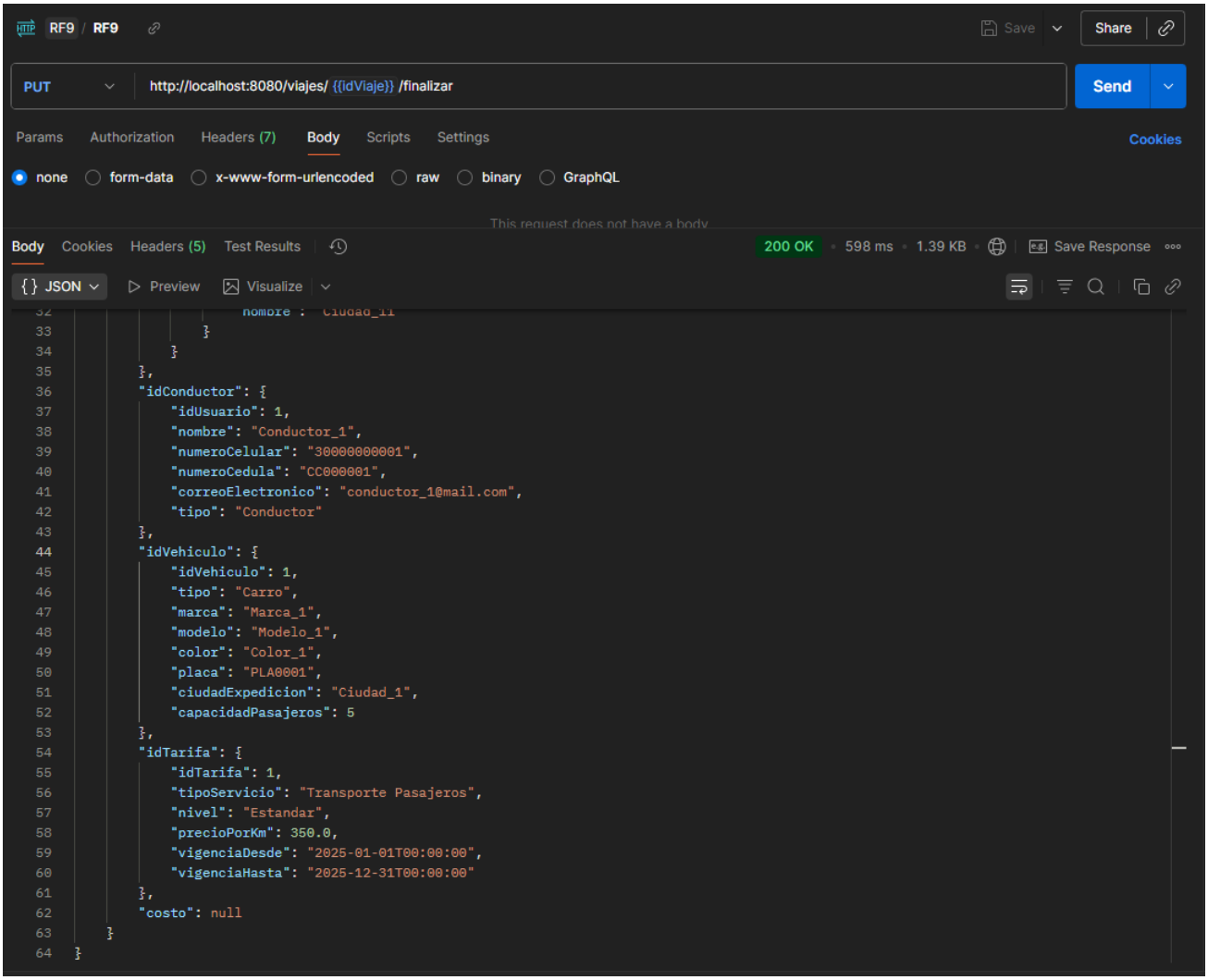


RF8: Solicitar un servicio por parte de un usuario de servicios

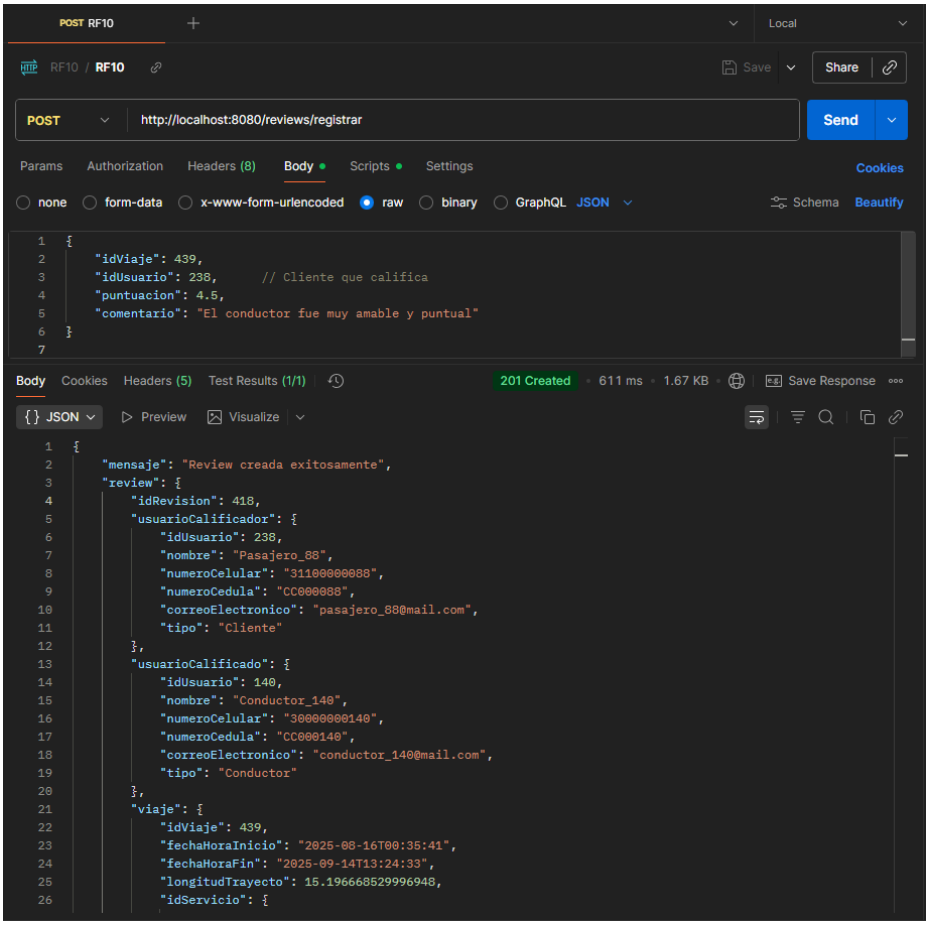


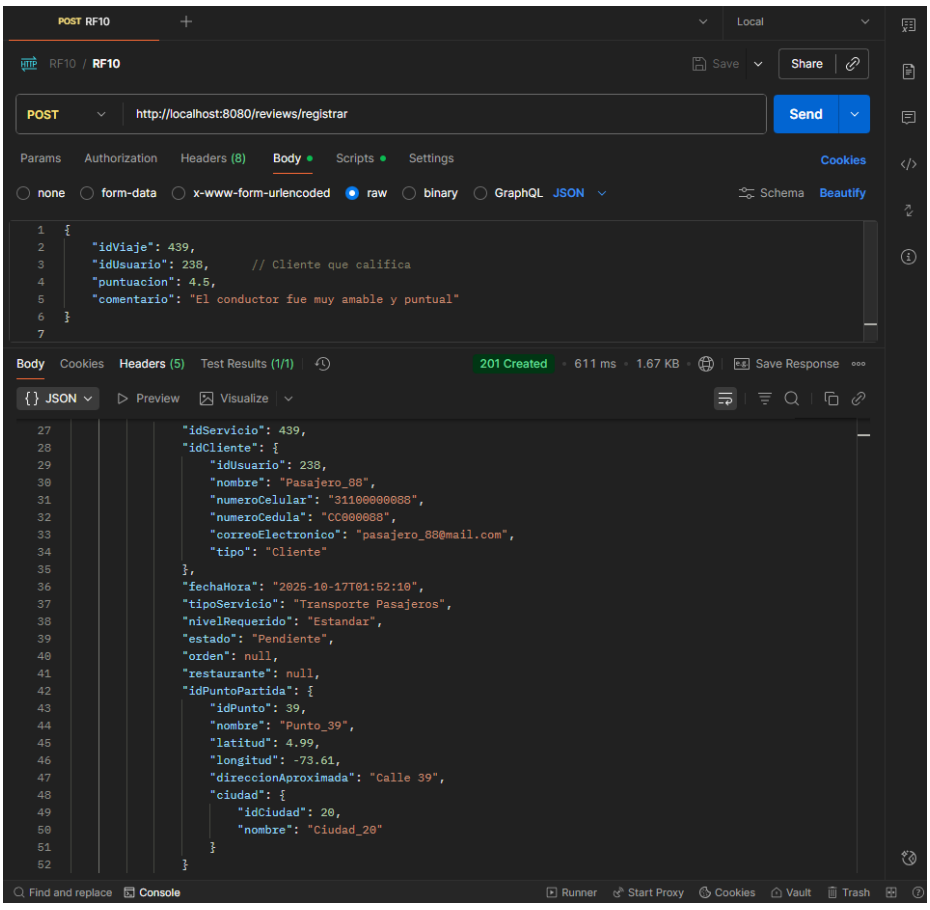
RF9: Registrar el final de un viaje para un usuario de servicios y un usuario conductor

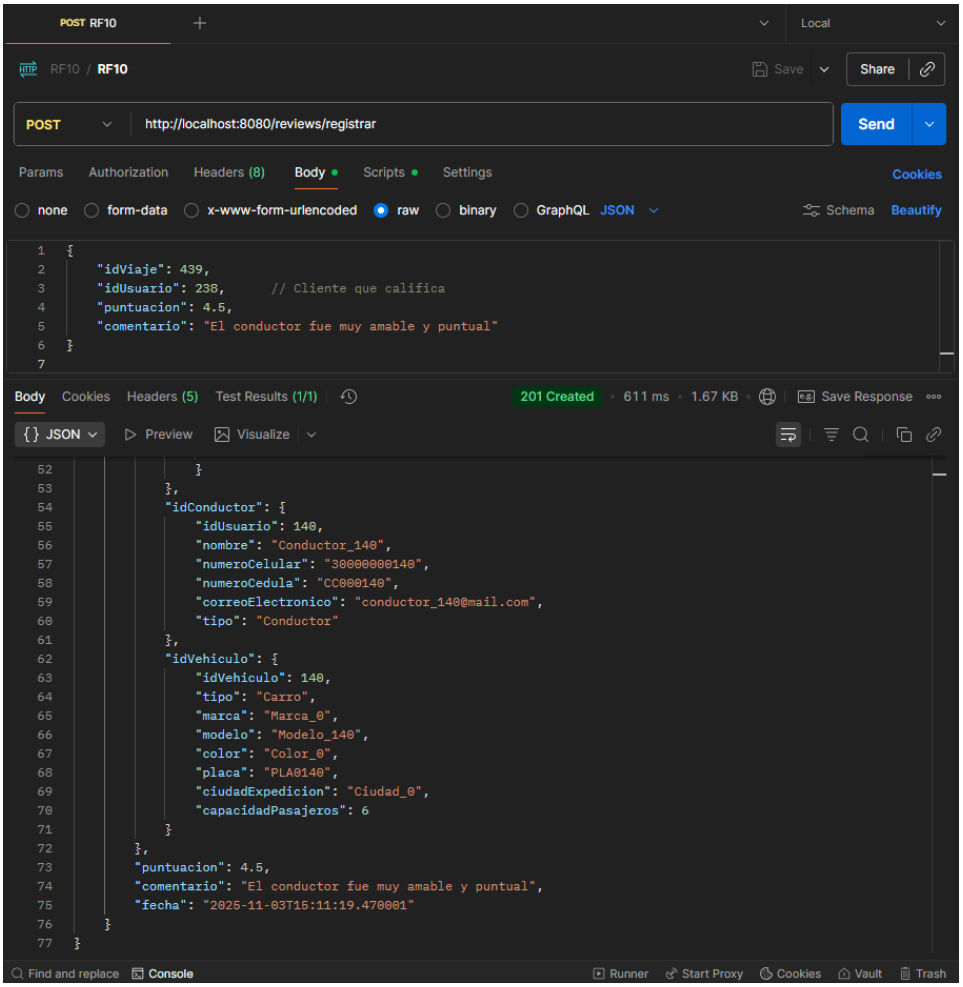

RF9 / RF9
Save
Share
PUT
http://localhost:8080/viajes/ {{idViaje}} /finalizar
Send
Params
Authorization
Headers (7)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
This request does not have a body
Body
Cookies
Headers (5)
Test Results
200 OK
598 ms
1.39 KB
Save Response
JSON
Preview
Visualize
{
"mensaje": "Viaje finalizado correctamente",
"viaje": {
"idViaje": 1259,
"fechaHoraInicio": "2025-11-03T20:36:39.555457",
"fechaHoraFin": "2025-11-03T20:36:45.3125473",
"longitudTrayecto": 10.988967953550775,
"idServicio": {
"idServicio": 1259,
"idCliente": {
"idUsuario": 251,
"nombre": "Pasajero_101",
"numeroCelular": "31100000101",
"numeroCedula": "CC000101",
"correoElectronico": "pasajero_101@mail.com",
"tipo": "Cliente"
},
"fechaHora": "2025-11-03T20:36:38.339582",
"tipoServicio": "Transporte Pasajeros",
"nivelRequerido": "Estandar",
"estado": "Finalizado",
"orden": null,
"restaurante": null,
"idPuntoPartida": {
"idPunto": 10,
"nombre": "Punto_10",
"latitud": 4.7,
"longitud": -73.9,
"direccionAproximada": "Calle 10",
"ciudad": {
"idCiudad": 11,
"nombre": "Ciudad_11"



RF10: Dejar una revisión por parte del usuario de servicios

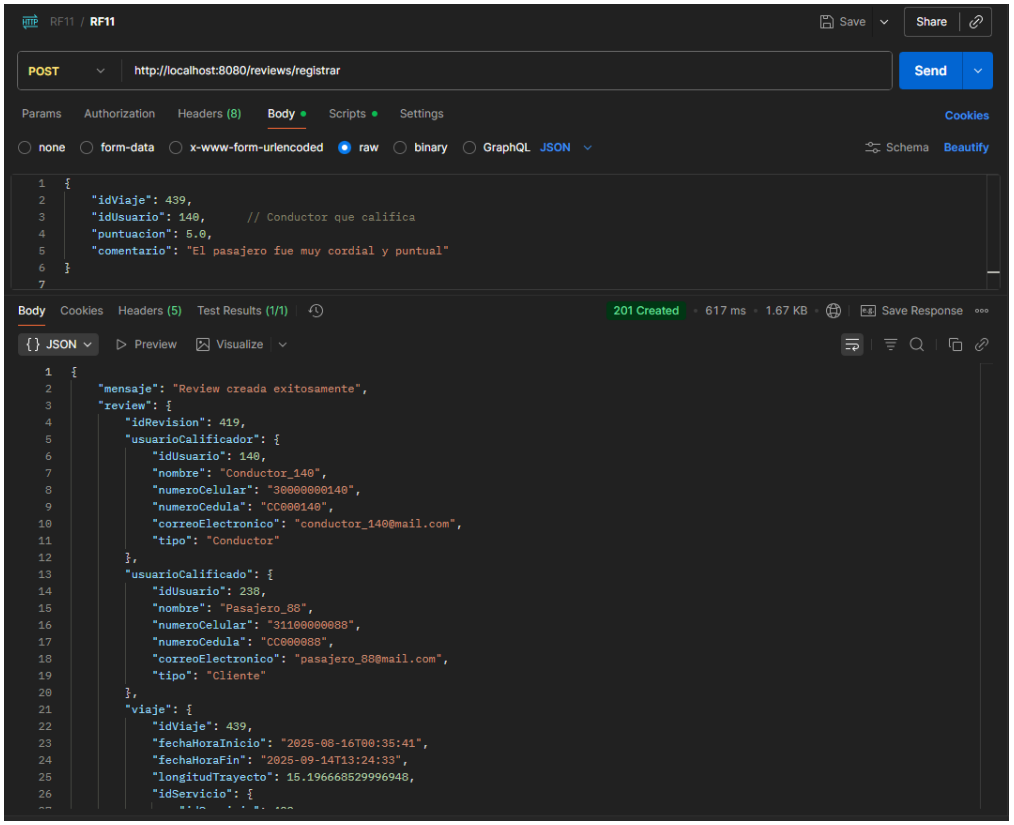

POST RF10
Local
RF10 / RF10
Save
Share
POST
http://localhost:8080/reviews/registrar
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Schema
Beautify
{
"idViaje": 439,
"idUsuario": 238, // Cliente que califica
"puntuacion": 4.5,
"comentario": "El conductor fue muy amable y puntual"
}
Body
Cookies
Headers (5)
Test Results (1/1)
201 Created
611 ms
1.67 KB
Save Response
JSON
Preview
Visualize
{
"mensaje": "Review creada exitosamente",
"review": {
"idRevision": 418,
"usuarioCalificador": {
"idUsuario": 238,
"nombre": "Pasajero_88",
"numeroCelular": "31100000088",
"numeroCedula": "CC000088",
"correoElectronico": "pasajero_88@mail.com",
"tipo": "Cliente"
},
"usuarioCalificado": {
"idUsuario": 140,
"nombre": "Conductor_140",
"numeroCelular": "30000000140",
"numeroCedula": "CC000140",
"correoElectronico": "conductor_140@mail.com",
"tipo": "Conductor"
},
"viaje": {
"idViaje": 439,
"fechaHoraInicio": "2025-08-16T00:35:41",
"fechaHoraFin": "2025-09-14T13:24:33",
"longitudTrayecto": 15.196668529996948,
"idServicio": {

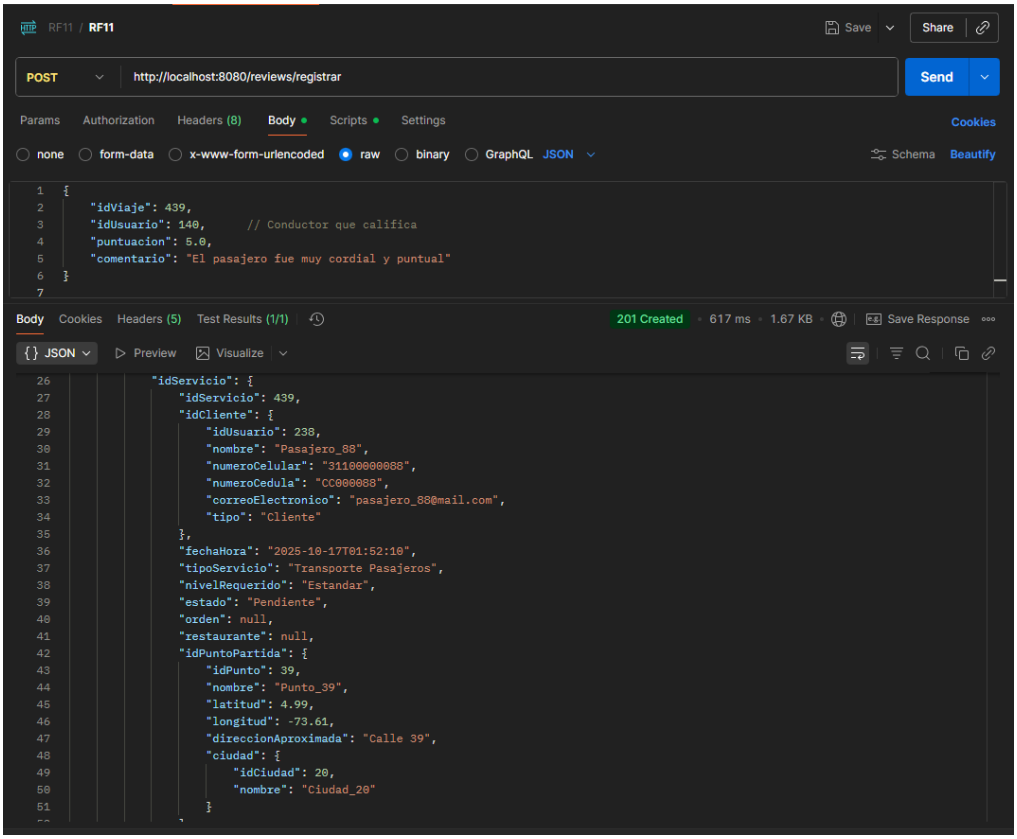

POST RF10
Local
RF10 / RF10
Save
Share
POST
http://localhost:8080/reviews/registrar
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Schema
Beautify
{
"idViaje": 439,
"idUsuario": 238, // Cliente que califica
"puntuacion": 4.5,
"comentario": "El conductor fue muy amable y puntual"
}
Body
Cookies
Headers (5)
Test Results (1/1)
201 Created
611 ms
1.67 KB
Save Response
JSON
Preview
Visualize
"idServicio": 439,
"idCliente": {
"idUsuario": 238,
"nombre": "Pasajero_88",
"numeroCelular": "31100000088",
"numeroCedula": "CC000088",
"correoElectronico": "pasajero_88@mail.com",
"tipo": "Cliente"
},
"fechaHora": "2025-10-17T01:52:10",
"tipoServicio": "Transporte Pasajeros",
"nivelRequerido": "Estandar",
"estado": "Pendiente",
"orden": null,
"restaurante": null,
"idPuntoPartida": {
"idPunto": 39,
"nombre": "Punto_39",
"latitud": 4.99,
"longitud": -73.61,
"direccionAproximada": "Calle 39",
"ciudad": {
"idCiudad": 20,
"nombre": "Ciudad_20"
}
}
Find and replace
Console
Runner
Start Proxy
Cookies
Vault
Trash

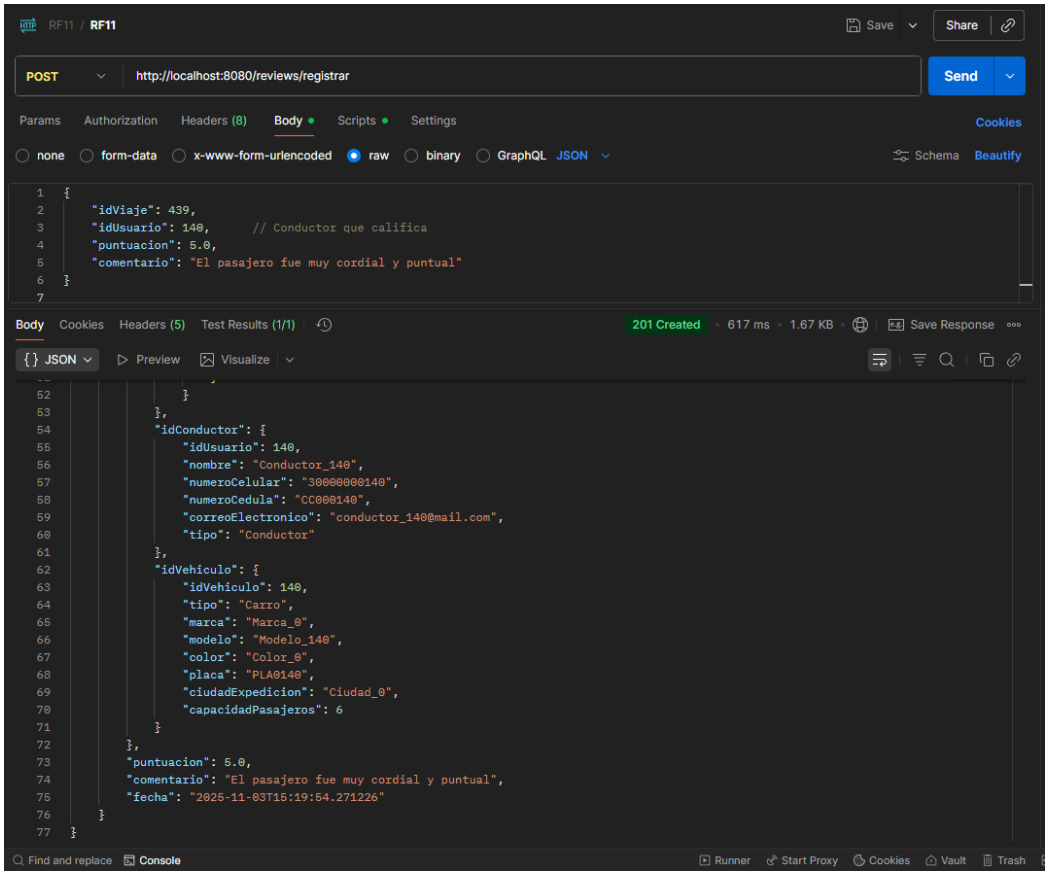


	IDREVISION	IDUSUARIOCALIFICADOR	IDUSUARIOCALIFICADO	PUNTUACION	COMENTARIO	FECHA	IDVIAJE
1	417	238	140	4.5	El conductor fue muy amable y puntual	2025-11-03T15:04:30.478640	439

RF11: Dejar una revisión por parte de un usuario conductor


RF11 / RF11
Save
Share
POST
http://localhost:8080/reviews/registrar
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Schema
Beautify
{
"idViaje": 439,
"idUsuario": 140, // Conductor que califica
"puntuacion": 5.0,
"comentario": "El pasajero fue muy cordial y puntual"
}
Body
Cookies
Headers (5)
Test Results (1/1)
201 Created
617 ms
1.67 KB
Save Response
JSON
Preview
Visualize
{
"mensaje": "Review creada exitosamente",
"review": {
"idRevision": 419,
"usuarioCalificador": {
"idUsuario": 140,
"nombre": "Conductor_140",
"numeroCelular": "3000000140",
"numeroCedula": "CC000140",
"correoElectronico": "conductor_140@mail.com",
"tipo": "Conductor"
},
"usuarioCalificado": {
"idUsuario": 238,
"nombre": "Pasajero_88",
"numeroCelular": "31100000088",
"numeroCedula": "CC000088",
"correoElectronico": "pasajero_88@mail.com",
"tipo": "Cliente"
},
"viaje": {
"idViaje": 439,
"fechaHoraInicio": "2025-08-16T00:35:41",
"fechaHoraFin": "2025-09-14T13:24:33",
"longitudTrayecto": 15.196668529996948,
"idServicio": {


RF11 / RF11
Save
Share
POST
http://localhost:8080/reviews/registrar
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Schema
Beautify
{
"idViaje": 439,
"idUsuario": 140, // Conductor que califica
"puntuacion": 5.0,
"comentario": "El pasajero fue muy cordial y puntual"
}
Body
Cookies
Headers (5)
Test Results (1/1)
201 Created
617 ms
1.67 KB
Save Response
JSON
Preview
Visualize
"idServicio": {
"idServicio": 439,
"idCliente": {
"idUsuario": 238,
"nombre": "Pasajero_88",
"numeroCelular": "31100000088",
"numeroCedula": "CC000088",
"correoElectronico": "pasajero_88@mail.com",
"tipo": "Cliente"
},
"fechaHora": "2025-10-17T01:52:10",
"tipoServicio": "Transporte Pasajeros",
"nivelRequerido": "Estandar",
"estado": "Pendiente",
"orden": null,
"restaurante": null,
"idPuntoPartida": {
"idPunto": 39,
"nombre": "Punto_39",
"latitud": 4.99,
"longitud": -73.61,
"direccionAproximada": "Calle 39",
"ciudad": {
"idCiudad": 20,
"nombre": "Ciudad_20"
}



419	419	140	238	5	El pasajero fue muy cordial y puntual	2025-11-03T15:19:54.271226	439

RFC1:

Tablas usadas: Usuarios, Servicio, Tarifa y viajes

Atributos usados: Usuarios: idUsuario

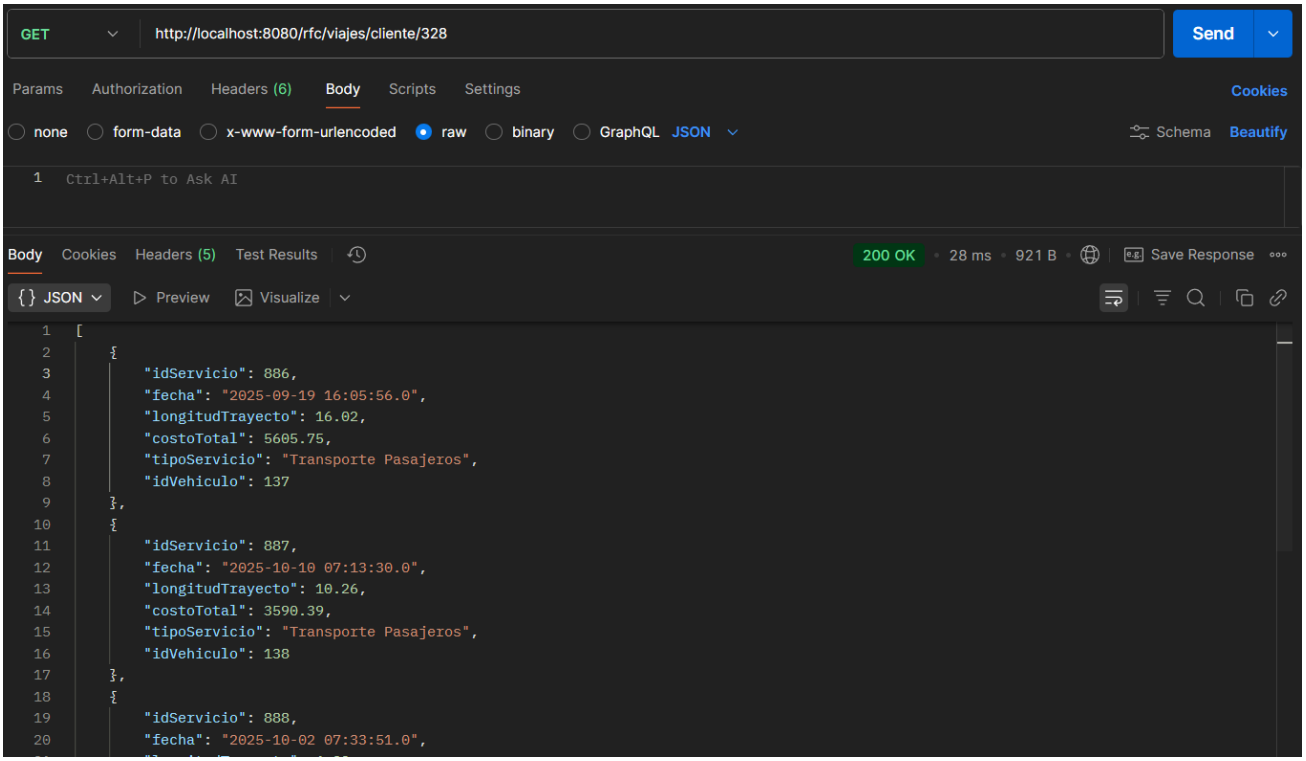
Servicios: idServicio, nivelRequerido

Tarifa: tipoServicio, nivel, vigenciaDesde, vigenciaHata

Vehiculo: fechaHoraInicio

JOIN: Se usó el INNER JOIN para en primera instancia unir el servicio junto con su usuario asignado, de viaje al servicio y tarifa a su tipo de servicio. Como los datos relevantes no contienen ningun valor nulo y solo usan datos en común usar este join era bastante apropiado.

```
---RFC 1
SELECT s.idServicio, v.fechaHoraInicio AS fecha, ROUND(v.longitudTrayecto, 2) AS longitudTrayecto,
ROUND((v.longitudTrayecto * t.precioPorKm), 2) AS costoTotal, s.tipoServicio, v.idVehiculo FROM usuarios du
INNER JOIN servicios s ON s.idCliente = du.idUsuario
INNER JOIN viajes v ON s.idServicio = v.idServicio
INNER JOIN tarifas t ON s.tipoServicio = t.tipoServicio AND t.nivel = s.nivelRequerido AND v.fechaHoraInicio
BETWEEN t.vigenciaDesde AND t.vigenciaHasta
WHERE du.idUsuario = 239;
```



RFC2:

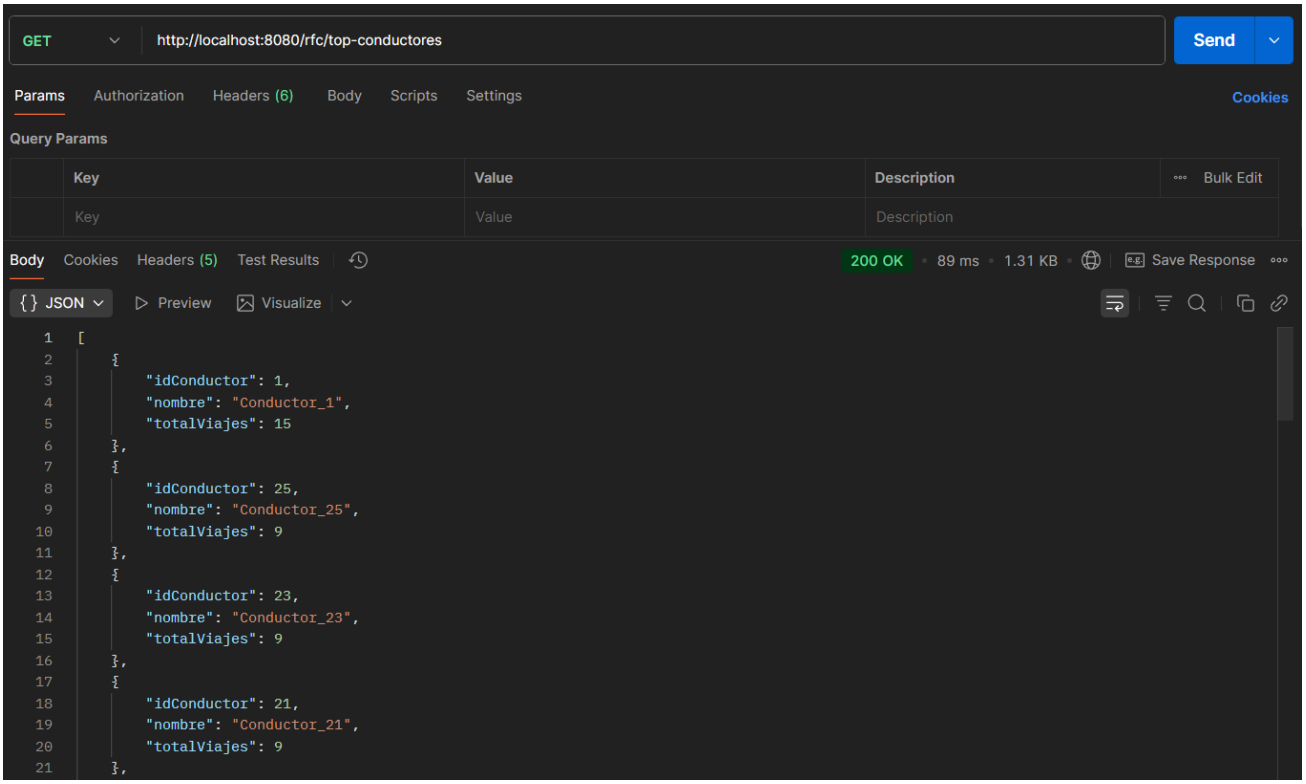
Tablas usadas: Usuarios y Vehículos

Atributos: Usuario: idUsuario, nombre

Viajes: idConductor

JOIN: Se uso INNER JOIN para unir Usuarios con vehículo, como siempre hay elementos en común, es apropiado usar este tipo de JOIN.

```
---RFC 2
SELECT u.idUsuario AS idConductor, u.nombre, COUNT(v.idViaje) AS total_viajes FROM viajes v
INNER JOIN usuarios u ON v.idConductor = u.idUsuario
GROUP BY u.idUsuario, u.nombre
ORDER BY total_viajes DESC
FETCH FIRST 20 ROWS ONLY;
```



RFC3:

Tablas usadas: Vehículo, Servicio y Tarifas

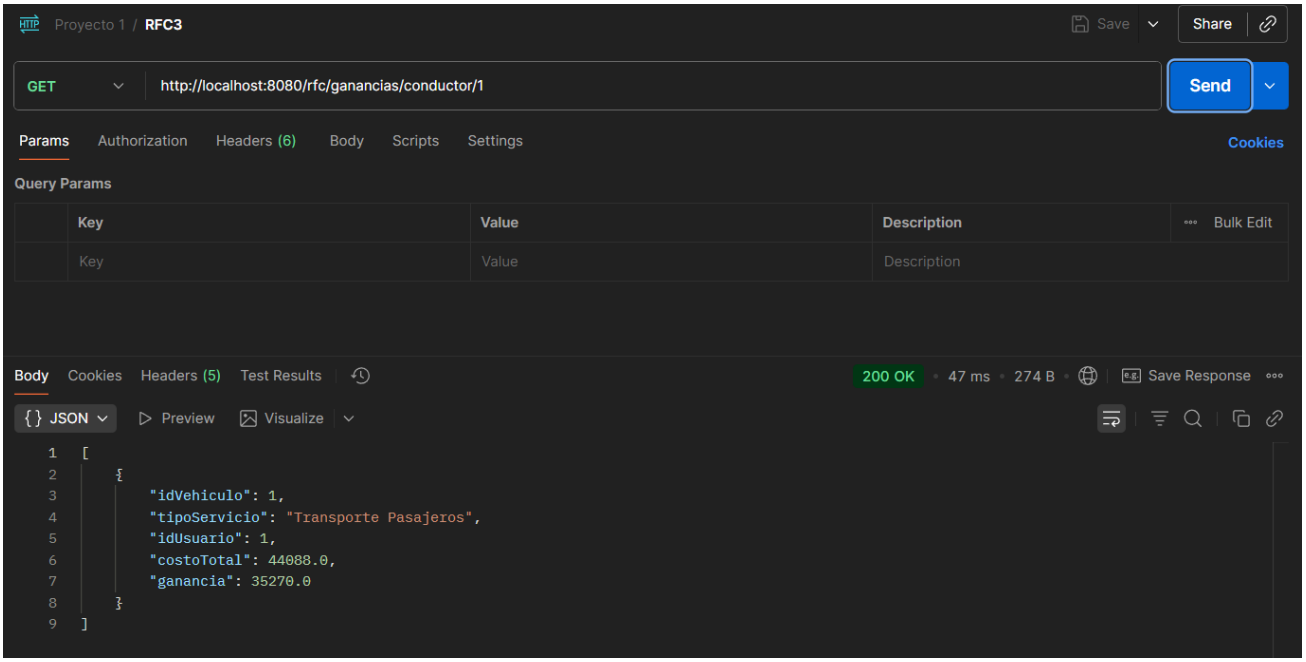
Atributos: Viajes: idVehiculo, idConductor, fechaHoraInicio, longitudTrayecto

Tarifa: precioPorKM, tipoServicio

Servicios: tipoServicio, idServicio, tipoServicio

JOIN: Siguiendo la misma lógica, al solo necesitar unir elementos en común, se usó INNER JOIN pues lo que se une como lo son los servicios con vehículos y servicios tarifa y servicios, no existe la posibilidad de que se aparezcan nulos.

```
---RFC 3
SELECT v.idVehiculo, s.tipoServicio, v.idConductor AS idUsuario, ROUND(SUM(v.longitudTrayecto * t.precioPorKm)) AS costo_total,
ROUND(SUM(v.longitudTrayecto * t.precioPorKm) * (1 - 0.2)) AS ganancia FROM viajes v
INNER JOIN servicios s ON v.idServicio = s.idServicio
INNER JOIN tarifas t ON s.tipoServicio = t.tipoServicio AND t.nivel = s.nivelRequerido
AND v.fechaHoraInicio BETWEEN t.vigenciaDesde AND t.vigenciaHasta
WHERE v.idConductor = 1
GROUP BY v.idVehiculo, s.tipoServicio, v.idConductor
ORDER BY v.idVehiculo, s.tipoServicio;
```



RFC4:

Tablas usadas: Ciudad, Servicio y PuntoUbicacion

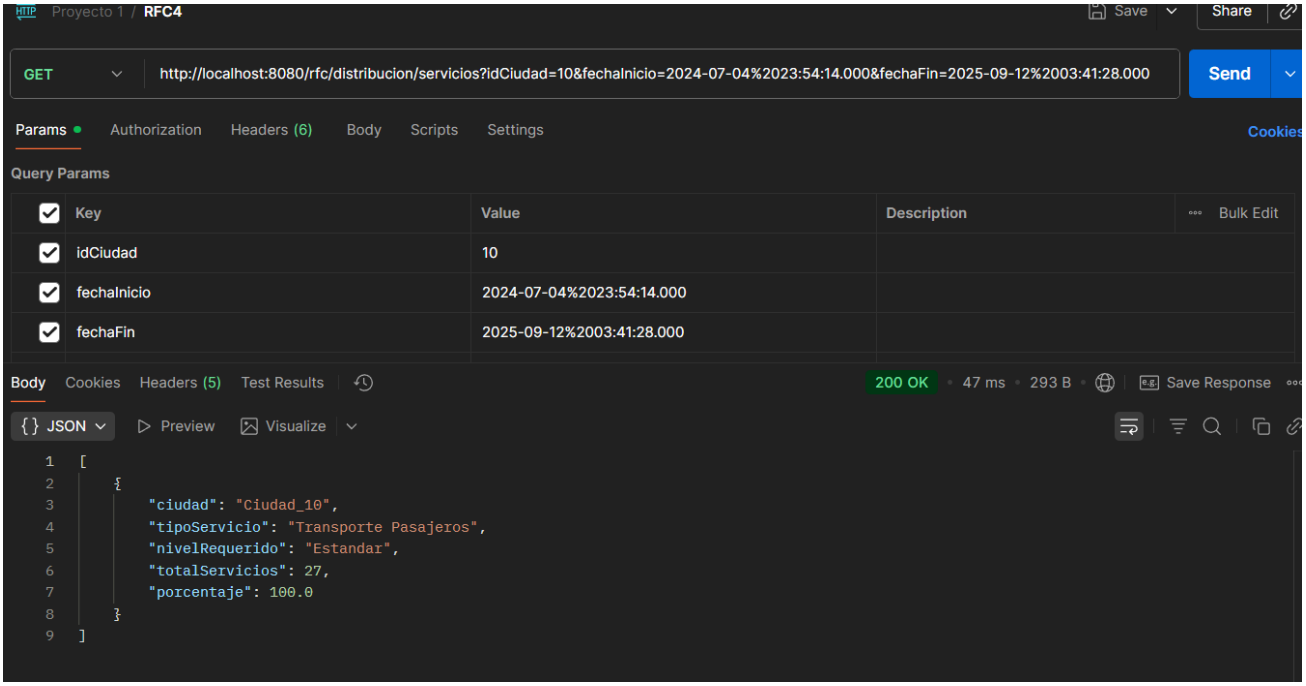
Atributos: Ciudad: nombre, idCiudad

Servicios: tipoServicios, nivelRequerido, id puntoPartida

PuntoPartida: idPunto, idCiudad

JOIN: Siguiendo la misma lógica, al solo necesitar unir elementos en común, se usó INNER JOIN.

```
---RFC 4
SELECT c.nombre AS ciudad, s.tipoServicio, s.nivelRequerido, COUNT(*) AS total_servicios,
 ROUND(100 * COUNT(*) / SUM(COUNT(*)) OVER (), 2) AS porcentaje FROM servicios s
INNER JOIN puntos_geograficos p ON s.idPuntoPartida = p.idPunto
INNER JOIN ciudades c ON p.idCiudad = c.idCiudad
WHERE c.idCiudad = 10 AND s.fechaHora BETWEEN TO_TIMESTAMP('2024-07-04 23:54:14.000','YYYY-MM-DD HH24:MI:SS.FF3')
AND TO_TIMESTAMP('2025-09-12 03:41:28.000','YYYY-MM-DD HH24:MI:SS.FF3')
GROUP BY c.nombre, s.tipoServicio, s.nivelRequerido
ORDER BY porcentaje DESC;
```

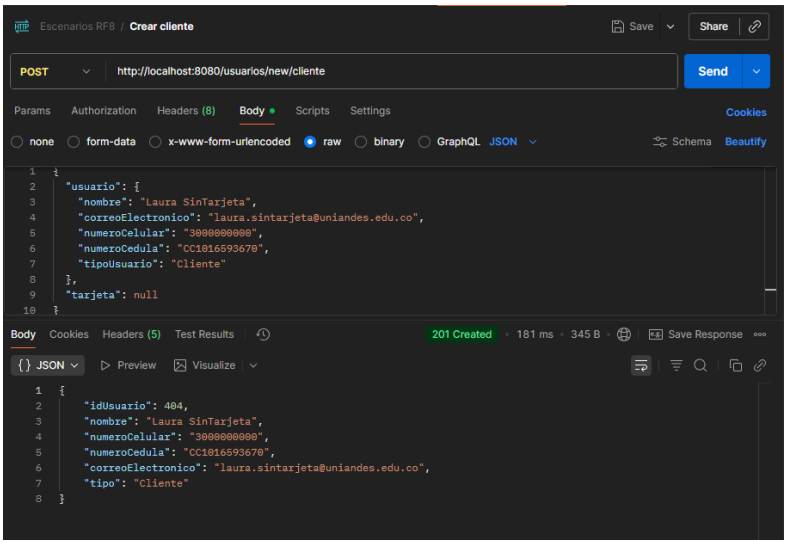


2. Implementación del RF8

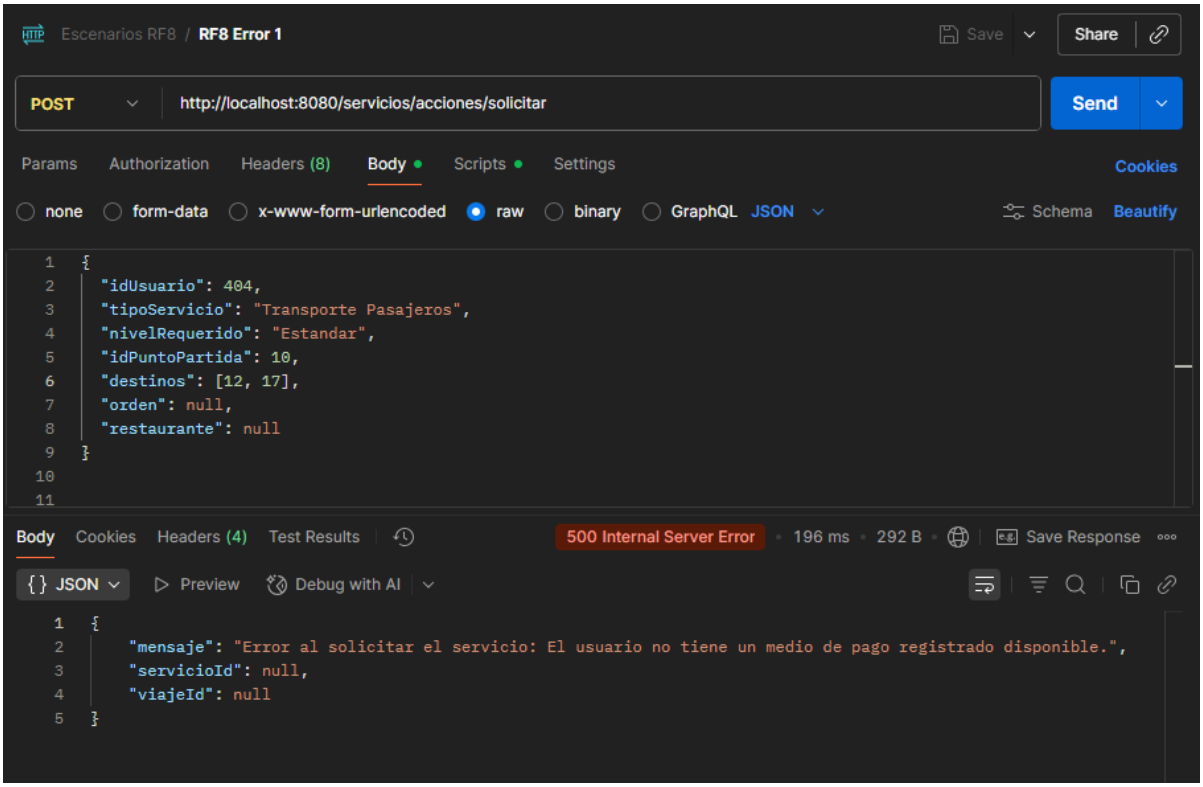
Se modelaron varios escenarios de prueba, esto con el fin de ver que en caso de fallas se generará un rollback de los cambios hechos hasta el momento. A continuación, se presentan los diferentes casos:

Escenario 1: Fallo porque el cliente no tiene método de pago

Primero, creamos un cliente sin tarjeta:



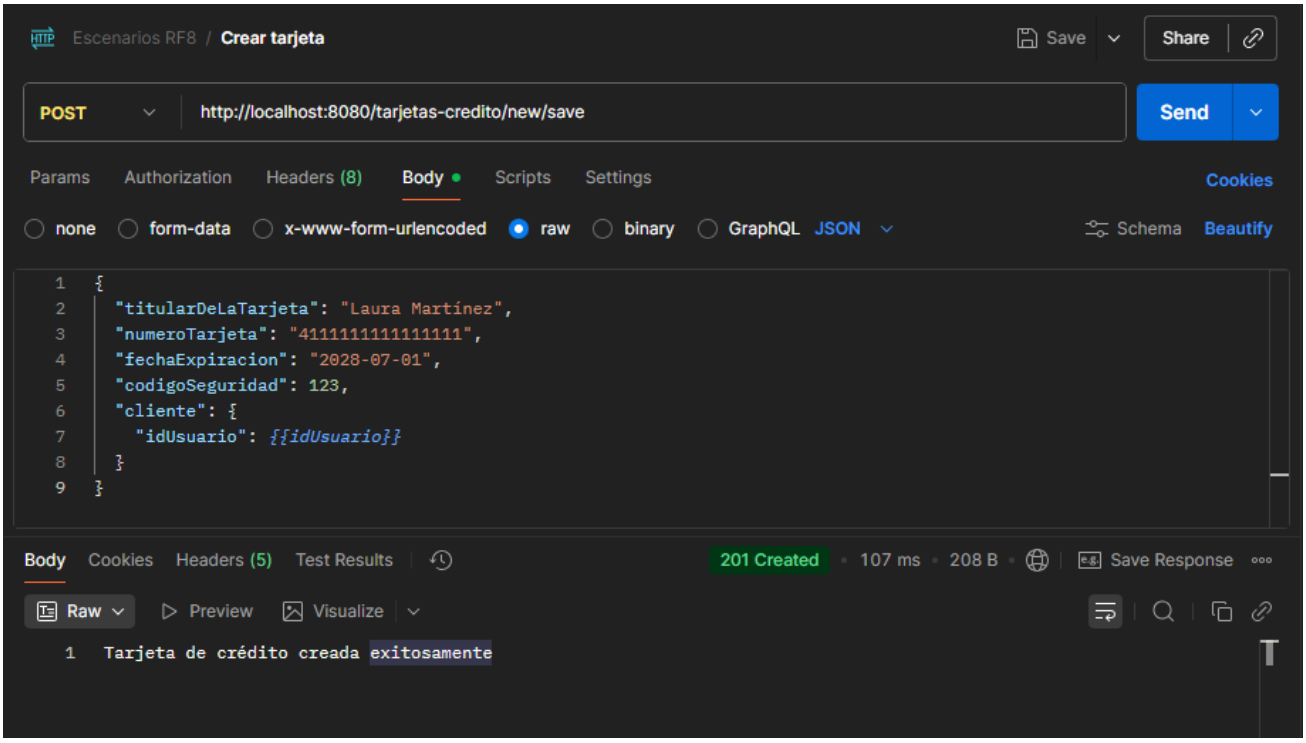
Ahora, intentamos que dicho cliente solicite un servicio para ver qué pasa:

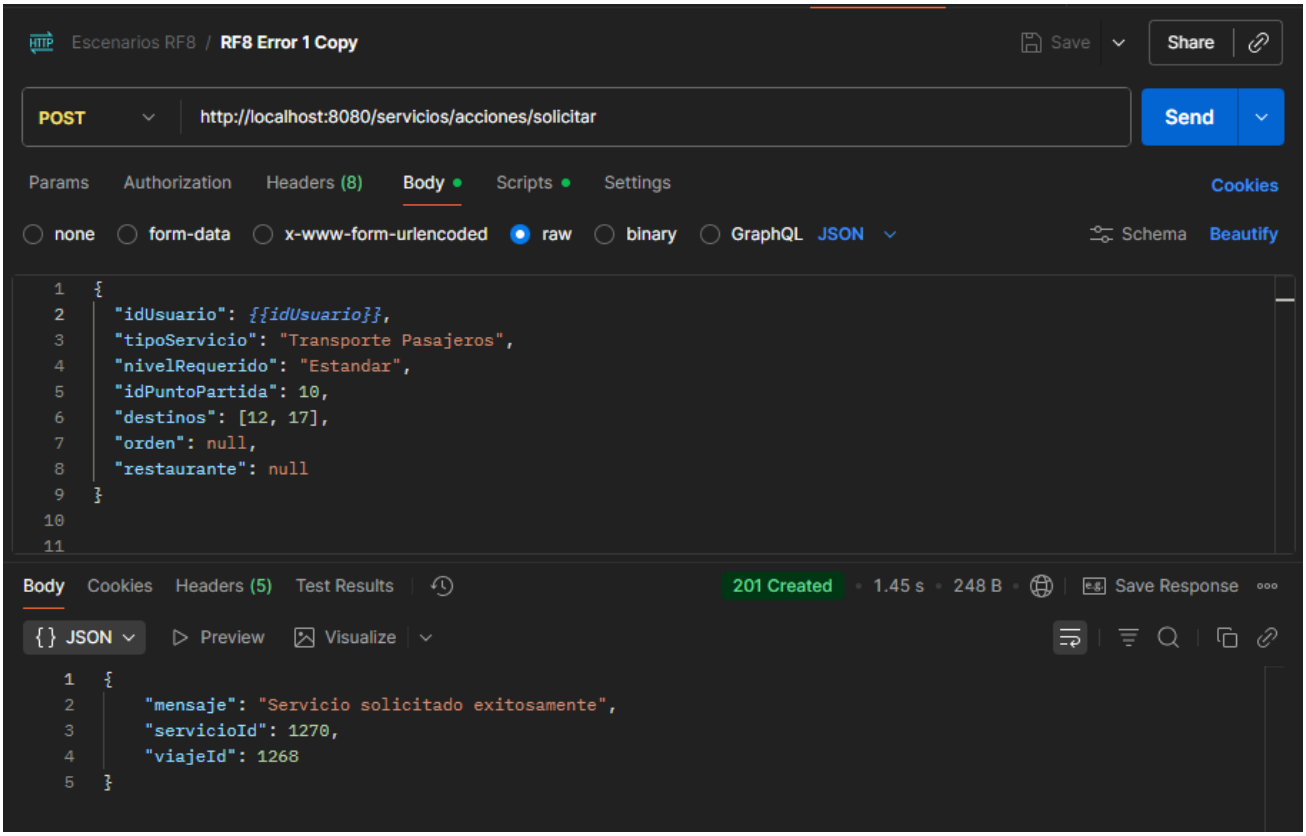


Como se observa, se genera error porque el cliente no tiene método de pago registrado y, por ende, no puede solicitar un servicio.

Escenario 2: Éxito cuando el cliente tiene método de pago

Ahora si se asocia un método de pago al cliente, por lo que ya no fallará por eso:

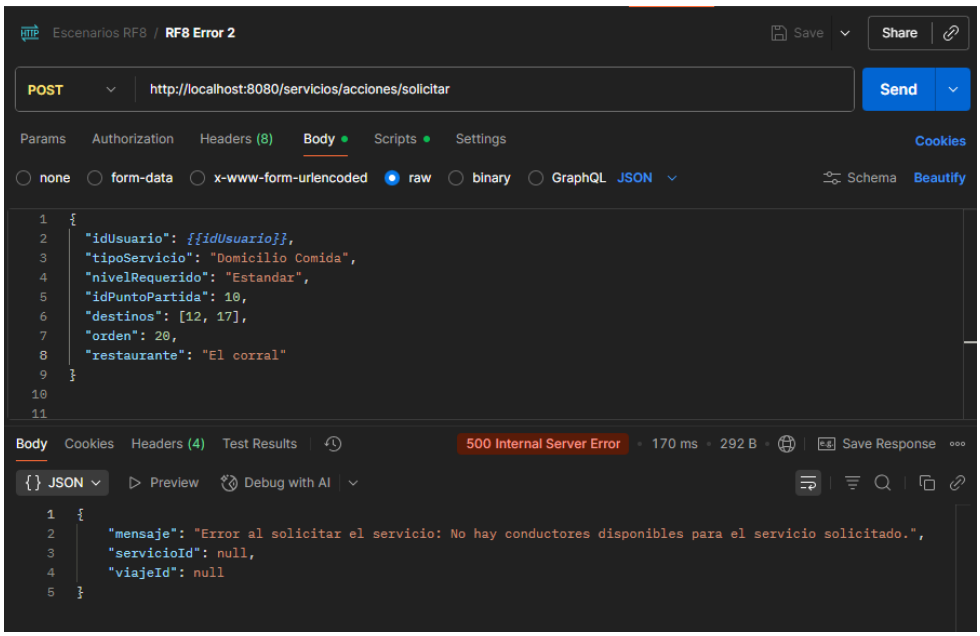




Como vemos, el usuario logra solicitar un servicio correctamente.

Escenario 2: Fallo cuando no hay conductores disponibles para dar el servicio

Para modelar esto, escogemos “Domicilio Comida” como el tipo de servicio ya que no hay muchos conductores ni vehículos en esa categoría:



Como se observa, no es posible asignar un conductor al servicio solicitado debido a la falta de disponibilidad, por lo que se hace rollback.

Escenario 3: Fallo cuando hay pocos conductores para el servicio y todos están ocupados

Para modelar esto, vamos a solicitar muchos servicios en la misma franja horaria hasta que eventualmente ya no haya disponibilidad:

	IDVIAJE	FECHAHORAINICIO	FECHAHORAFIN	LONGITUDTRAYECTO	IDSERVICIO	IDCONDUCTOR	IDVEHICULO	IDTARIFA	COSTO
1	1281	2025-11-03T22:05:33.003812	(null)	(null)	1284	1	1	1	(null)
2	1282	2025-11-03T22:05:35.387473	(null)	(null)	1285	1	1	1	(null)
3	1283	2025-11-03T22:05:37.959969	(null)	(null)	1286	1	1	1	(null)
4	1284	2025-11-03T22:05:40.564663	(null)	(null)	1287	1	1	1	(null)
5	1285	2025-11-03T22:05:43.407933	(null)	(null)	1288	1	1	1	(null)
6	1286	2025-11-03T22:05:45.765905	(null)	(null)	1289	1	1	1	(null)
7	1287	2025-11-03T22:05:48.155566	(null)	(null)	1290	1	1	1	(null)
8	1288	2025-11-03T22:05:50.798101	(null)	(null)	1291	1	1	1	(null)
9	1289	2025-11-03T22:05:59.526333	(null)	(null)	1292	1	1	1	(null)
10	1290	2025-11-03T22:06:02.576801	(null)	(null)	1293	1	1	1	(null)

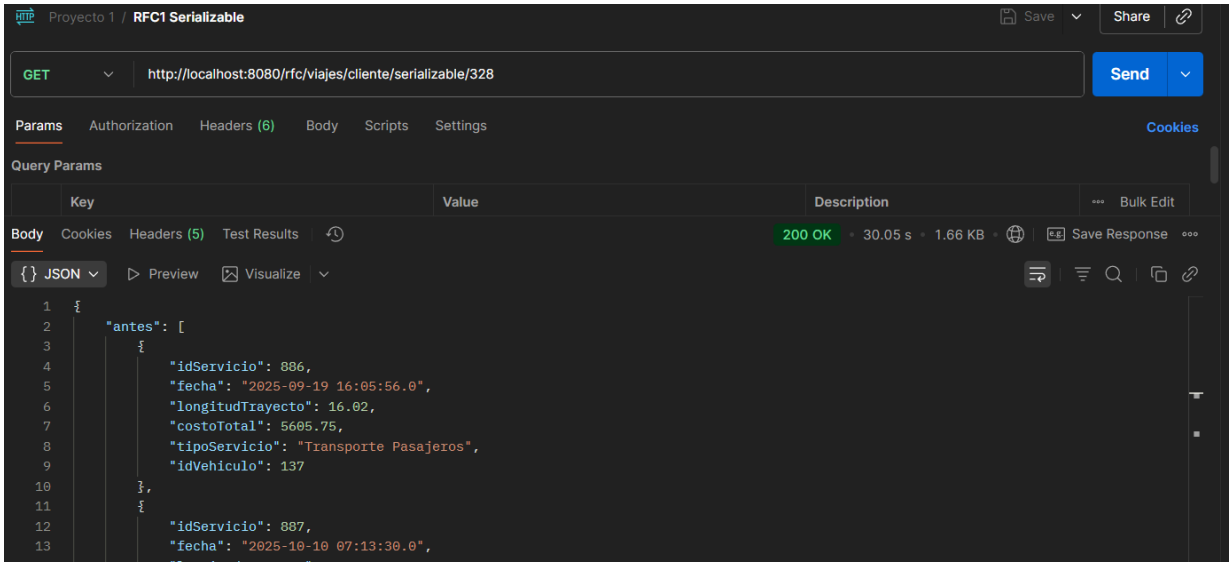
Como se observa, por alguna razón, no funciona como se espera, no está actualizando correctamente la disponibilidad, por lo que siempre puede coger al conductor 1 para esta franja.

	IDVIAJE	FECHAHORAINICIO	FECHAHORAFIN	LONGITUDTRAYECTO	IDSERVICIO	IDCONDUCTOR	IDVEHICULO	IDTARIFA	COSTO
9	1309	2025-11-03T22:24:26.160232	(null)	(null)	1312	3	3	1	(null)
10	1310	2025-11-03T22:24:29.254777	(null)	(null)	1313	3	3	1	(null)
11	1311	2025-11-03T22:24:32.190528	(null)	(null)	1314	4	4	1	(null)
12	1312	2025-11-03T22:24:36.139941	(null)	(null)	1315	4	4	1	(null)
13	1313	2025-11-03T22:24:39.465053	(null)	(null)	1316	4	4	1	(null)
14	1314	2025-11-03T22:24:42.744720	(null)	(null)	1317	5	5	1	(null)
15	1315	2025-11-03T22:24:46.455761	2025-11-03T22:34:29.411685	10.988967953550775	1318	5	5	1	3846.13878374277
16	1316	2025-11-03T22:24:49.502493	(null)	(null)	1319	5	5	1	(null)
17	1317	2025-11-03T22:24:52.502297	2025-11-03T22:25:21.635622	10.988967953550775	1320	6	6	1	3846.13878374277
18	1318	2025-11-03T22:24:55.354338	2025-11-03T22:25:09.710079	10.988967953550775	1321	6	6	1	3846.13878374277
19	1319	2025-11-03T22:33:19.058065	(null)	(null)	1323	6	6	1	(null)
20	1320	2025-11-03T22:33:24.065962	(null)	(null)	1324	6	6	1	(null)
21	1321	2025-11-03T22:33:29.796437	(null)	(null)	1325	6	6	1	(null)
22	1322	2025-11-03T22:33:34.427918	2025-11-03T22:34:16.631526	10.988967953550775	1326	7	7	1	3846.13878374277

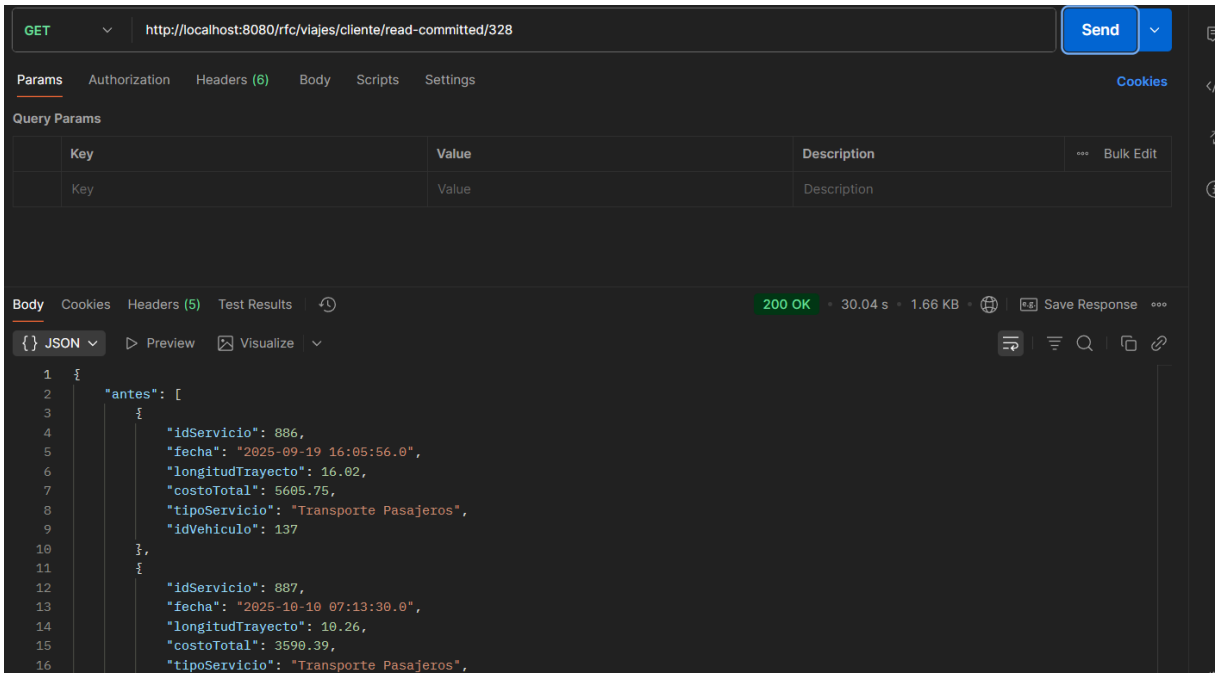
3. **Implementación del RFC1 con los niveles de aislamiento con read-committed y serializable.**

   Al requerimiento funcional de consulta 1 se le crearon dos versiones de este, cada una con un nivel de aislamiento diferente, esto cambiando el parámetro isolation al darle la transaccionalidad en el service de los RFC. A estas se les agrego un tiempo de espera tal y como fue solicitado en el enunciado.

   Serializable:



Read Committed:



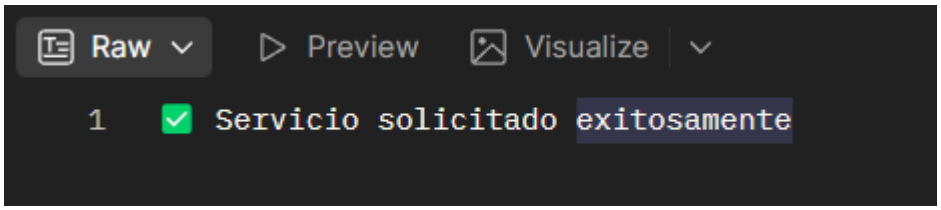
4. Concurrencia Serializable

Línea de tiempo: T1 corresponde a la transacción del RFC1 y T2 a la del RF8

1) T1: Consulta el histórico de viajes de usuarios, estos son los últimos resultados de esta consulta.

```
{
    "idServicio": 1262,
    "fecha": "2025-11-03 20:40:16.921226",
    "longitudTrayecto": null,
    "costoTotal": null,
    "tipoServicio": "Transporte Pasajeros",
    "idVehiculo": 1
},
{
    "idServicio": 1263,
    "fecha": "2025-11-03 20:40:55.160644",
    "longitudTrayecto": null,
    "costoTotal": null,
    "tipoServicio": "Transporte Pasajeros",
    "idVehiculo": 1
}
```


2) T2: el mismo usuario solicita un viaje.



3) T3: Se repite la consulta del histórico de viajes de un usuario


```
{
    "idServicio": 1262,
    "fecha": "2025-11-03 20:40:16.921226",
    "longitudTrayecto": null,
    "costoTotal": null,
    "tipoServicio": "Transporte Pasajeros",
    "idVehiculo": 1
},
{
    "idServicio": 1263,
    "fecha": "2025-11-03 20:40:55.160644",
    "longitudTrayecto": null,
    "costoTotal": null,
    "tipoServicio": "Transporte Pasajeros",
    "idVehiculo": 1
}
```


En este escenario, a pesar de que la transacción del RF8 se ejecutó con normalidad sin tener que esperar a que termine la otra, la transacción del RFC1 mantuvo los datos tanto del antes como del después de haber esperado los 30 segundos. Esto evidenciando que, gracias a su nivel de aislamiento, evitó una lectura no repetible y manteniendo coherencia en los datos en todo momento de la transacción del RFC1.

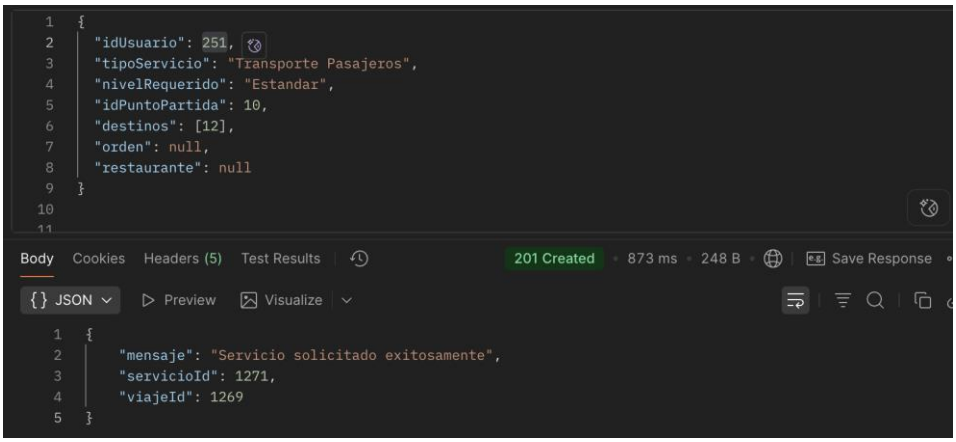
5. Concurrencia Read Commited

Línea de tiempo: T1 corresponde a la transacción del RFC1 y T2 a la del RF8

1) T1: Consulta el histórico de viajes de usuarios, estos son los últimos resultados de esta consulta.

```
{
    "idServicio": 1264,
    "fecha": "2025-11-03 20:43:37.876108",
    "longitudTrayecto": null,
    "costoTotal": null,
    "tipoServicio": "Transporte Pasajeros",
    "idVehiculo": 1
},
{
    "idServicio": 1267,
    "fecha": "2025-11-03 21:53:11.38192",
    "longitudTrayecto": null,
    "costoTotal": null,
    "tipoServicio": "Transporte Pasajeros",
    "idVehiculo": 1
}
```


2) T2: el mismo usuario solicita un viaje.



3) T3: Se repite la consulta del histórico de viajes de un usuario. Los últimos en el listado se encuentran a continuación.


```
{
    "idServicio": 1267,
    "fecha": "2025-11-03 21:53:11.38192",
    "longitudTrayecto": null,
    "costoTotal": null,
    "tipoServicio": "Transporte Pasajeros",
    "idVehiculo": 1
},
{
    "idServicio": 1271,
    "fecha": "2025-11-03 21:59:55.618884",
    "longitudTrayecto": null,
    "costoTotal": null,
    "tipoServicio": "Transporte Pasajeros",
    "idVehiculo": 1
}
```


Se realizó una consulta del histórico de viajes (T1) y, tras una espera de 30 segundos, se repitió la misma consulta (T3). Entre ambos instantes, se ejecutó el RF8, solicitando un nuevo servicio,

el cual generó un nuevo registro en la tabla de viajes. Dado el nivel de aislamiento READ COMMITED, la segunda consulta de RFC1 tuvo acceso a los cambios que ya habían sido confirmados por la transacción RF8. Por ello, el viaje solicitado entre T1 y T3 aparece reflejado únicamente en el segundo resultado. Con ello en mente, se afirma que RFC1 no debió esperar a que terminara la ejecución para ver el resultado de RF8, ya que el nivel de aislamiento configurado permite la lectura de datos confirmados por otras transacciones una vez estas dinalizan, sin bloquear ni retrasar la ejecución de la consulta inicial.