

Entrega 3 AlpesCab

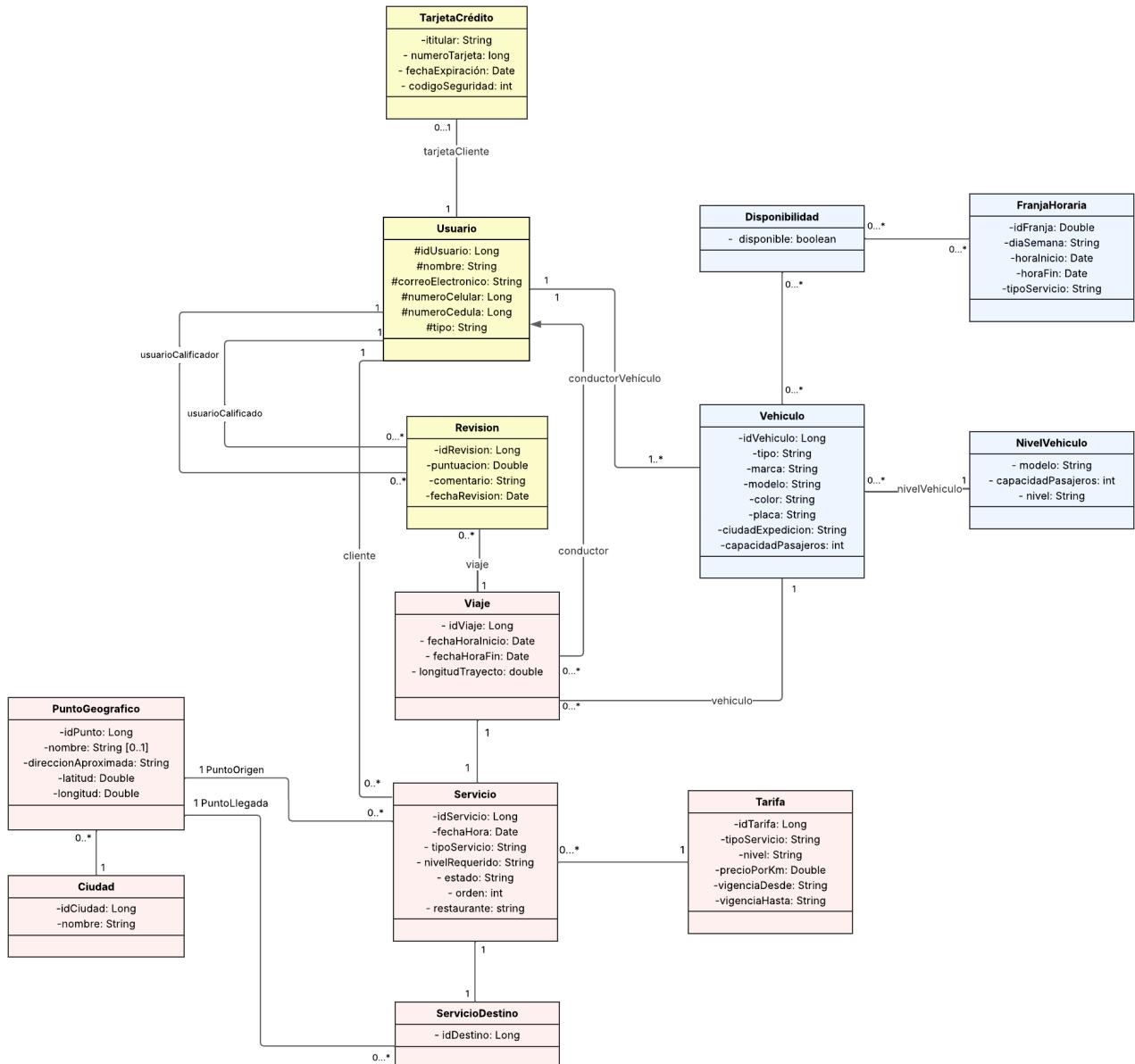
Sergio Arias | 202412370

Sara González | 202210908

Laura Martinez-Galindo | 202125643

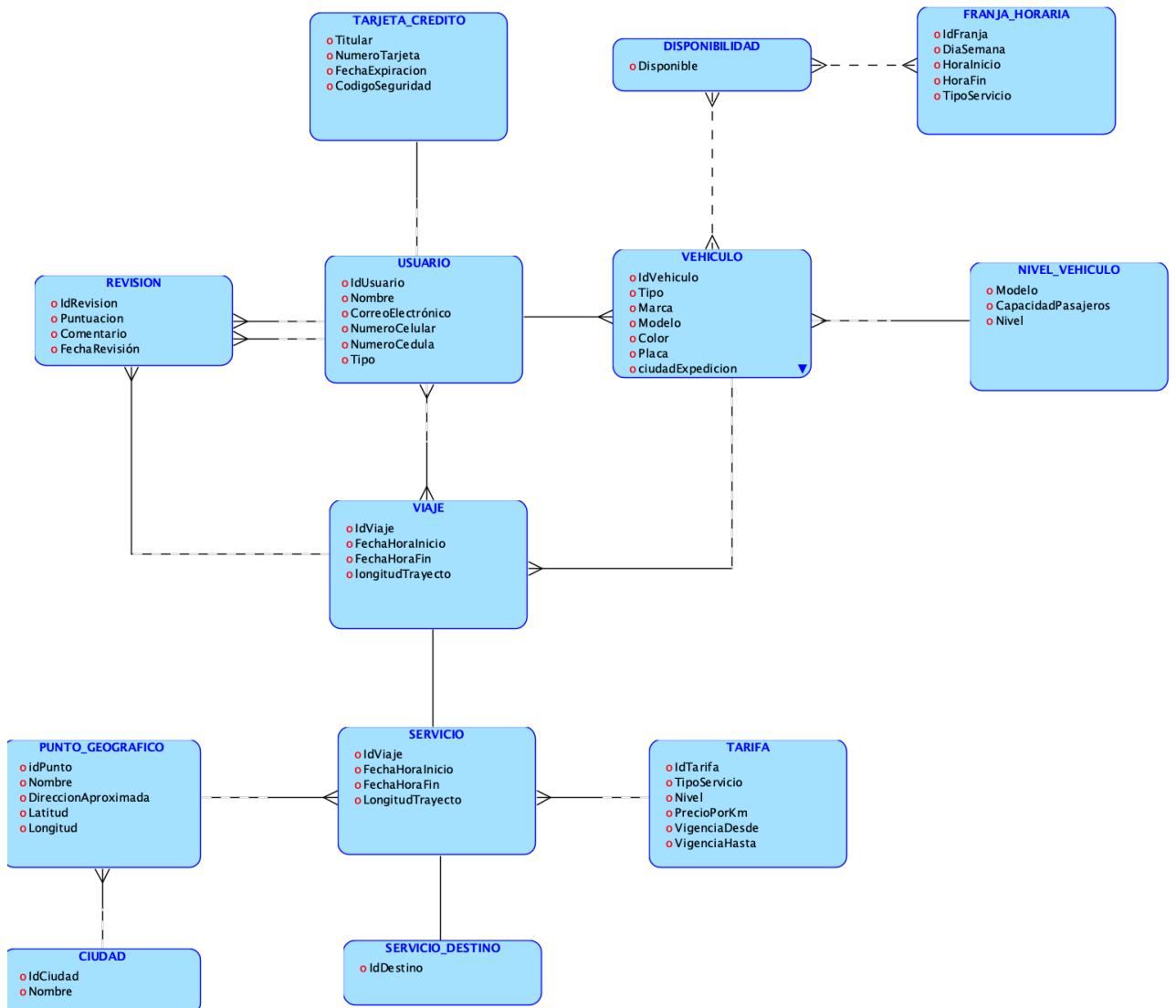
0. Modelamiento de AlpesCab

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@gmail.com	cliente
2	Tatiana	3109876543	1023456781	nierfan@gmail.com	conductor
3	Carlos	3206549871	1034567892	9sbszoh@gmail.com	cliente
4	Luka	3156781234	1045678912	LukaAnchor@gmail.com	cliente
5	Stinson	3155755237	1056789123	barney939@gmail.com	conductor
6	Robin	3112345678	1105912091	nightwing771@gmail.com	cliente

Review

idRevision	idUsuarioCalificador	idUsuarioCalificado	puntuacion	comentario	fecha	idViaje
PK, NN, SA, ND	FK(User.idUsuario), NN, CK	FK(User.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 semáforos	8/31/2025	4
5	1	1	1	Iba viendo tiktok manejando	8/31/2025	5

ConductorVehiculo

idConductor	idVehiculo
PK, FK(User.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(User.idUsuario), CK(User.tipo="Cliente")
1	Barney Simon	123456789	27/11	245	1
2	MADIE DE TATIANA	987654321	27/01	111	2
3	Dick Grayson	123769456	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	HU123	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	4.6104	-74.070288	Carrera 7 con Calle 24	1
2	NULL	4.591053	-74.072167	nro Carreras Séptima y Octava con Calles Décima	1
3	Parque Avi	4.65	-74.1	En la localidad de Teusaquillo, atravesado por las carrer	1
4	NULL	4.53604	-74.1388	Carrera 5 #12-41	1
5	NULL	0	0	Centro	5

Ciudad		
idCiudad	PK	nombre
	1	Bogotá
	2	Cali
	3	Medellín
	4	Barranquilla
	5	Santa Marta

Servicio									
idServicio	idCliente	fechaHora	tipoServicio	nivelFrecuendo	estado	orden	restaurante	idPuntoPanida	
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	23/03/2025 9:05	Transporte Pasajeros	Estandar	Pendiente	NULL	NULL	1	
2	5	23/03/2025 9:05	Domicilio Comida	Confort	Asignado	34	El Cedral	2	
3	6	23/03/2025 9:05	Transporte Paquete	Large	Cancelado	NULL	NULL	3	
4	2	23/03/2025 9:05	Transporte Pasajeros	Confort	Pendiente	NULL	NULL	4	
5	5	23/03/2025 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

(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	3.0	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

The screenshot shows a Postman collection named 'RF1 / RF1'. A POST request is made to `http://localhost:8080/ciudades/new/save`. The body contains the following JSON:

```
1 {  
2   "nombre": "Ciudad_Prueba5"  
3 }  
4
```

The response is a 201 Created status with the following JSON body:

```
1 {  
2   "mensaje": "Ciudad creada exitosamente",  
3   "ciudad": {  
4     "idCiudad": 25,  
5     "nombre": "Ciudad_Prueba5"  
6   }  
7 }
```

RF2: Registrar un usuario de servicios

POST | http://localhost:8080/usuarios/new/cliente

Params Authorization Headers (8) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	...

Body Cookies Headers (5) Test Results | ⏪

201 Created | 427 ms | 325 B | Save Response | ⌂ | ⌂ | ⌂ | ⌂ | ⌂ | ⌂

{ } JSON | Preview | Visualize | ⌂ | ⌂ | ⌂ | ⌂ | ⌂ | ⌂

```
1 {  
2   "idUsuario": 401,  
3   "nombre": "Laura Martinez",  
4   "numeroCelular": "3001234567",  
5   "numeroCedula": "123456789",  
6   "correoElectronico": "laura@example.com",  
7   "tipo": "Cliente"  
8 }
```

RF3: Registrar un usuario conductor

HTTP RF3 / RF3

POST | http://localhost:8080/usuarios/new/conductor

Params Authorization Headers (8) Body Scripts Settings Cookies

None form-data x-www-form-urlencoded raw binary GraphQL JSON | Schema | Beautify

{ } JSON | Preview | Visualize | ⌂ | ⌂ | ⌂ | ⌂ | ⌂ | ⌂

Body Cookies Headers (5) Test Results (1/1) | ⏪

201 Created | 93 ms | 381 B | Save Response | ⌂ | ⌂ | ⌂ | ⌂ | ⌂ | ⌂

{ } JSON | Preview | Visualize | ⌂ | ⌂ | ⌂ | ⌂ | ⌂ | ⌂

```
1 {  
2   "mensaje": "Conductor creado exitosamente",  
3   "usuario": {  
4     "idUsuario": 402,  
5     "nombre": "Juan Pérez2",  
6     "numeroCelular": "3001234567",  
7     "numeroCedula": "123456789",  
8     "correoElectronico": "juan.perez@mail.com",  
9     "tipo": "Conductor"  
10    }  
11 }
```

RF4: Registrar un vehículo para un usuario conductor

HTTP RF4 / RF4

POST http://localhost:8080/vehiculos/new?conductorId=402

Params • Authorization Headers (8) Body • Scripts • Settings Cookies

None form-data x-www-form-urlencoded raw binary GraphQL JSON

Schema Beautify

```
1 {  
2   "tipo": "Carro",  
3   "marca": "Toyota",  
4   "modelo": "Corolla",  
5   "color": "Rojo",  
6   "placa": "ABC123",  
7   "ciudadExpedicion": "Bogotá",  
8   "capacidadPasajeros": 4  
9 }  
10
```

Body Cookies Headers (5) Test Results (1/1)

201 Created 549 ms 381 B Save Response

[] JSON ▾ Preview Visualize

```
1 {  
2   "mensaje": "Vehiculo registrado exitosamente",  
3   "vehiculo": {  
4     "idVehiculo": 151,  
5     "tipo": "Carro",  
6     "marca": "Toyota",  
7     "modelo": "Corolla",  
8     "color": "Rojo",  
9     "placa": "ABC123",  
10    "ciudadExpedicion": "Bogotá",  
11    "capacidadPasajeros": 4  
12  }  
13 }
```

151 151 Carro Toyota Corolla Rojo ABC123 Bogotá

151 402 151

RF5: Registrar la disponibilidad de un vehículo para servicios

HTTP New Collection / RF5

Save Share

POST localhost:8080/disponibilidades/registrar Send

Params Authorization Headers (8) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Schema Beautify

```
1 {  
2   "idConductor": 3,  
3   "diaSemana": "Lunes",  
4   "horaInicio": "08:00",  
5   "horaFin": "20:00",  
6   "tipoServicio": "Transporte"  
7 }
```

Body Cookies Headers (5) Test Results 200 OK 47 ms 203 B Save Response

Raw Preview Visualize Disponibilidad registrada correctamente

RF6: Modificar la disponibilidad de un vehículo para servicios

HTTP New Collection / RF6

POST http://localhost:8080/disponibilidades/modificar

Send

Params Authorization Headers (8) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Schema Beautify

```
1 {
2     "idConductor": 2,
3     "idVehiculoAnt": 1,
4     "idFranjaAnt": 3,
5     "diaSemana": "Lunes",
6     "horaInicio": "06:00",
7     "horaFin": "07:00",
8     "tipoServicio": "Transporte Pasajeros",
9     "idVehiculoNuevo": 1
10 }
11
```

Body Cookies Headers (5) Test Results 200 OK 225 ms 203 B Save Response

Raw Preview Visualize

```
1 Disponibilidad modificada correctamente
```

RF7: Registrar un punto geográfico

New Collection / RF7

POST http://localhost:8080/puntos-geograficos/new/save

Params Authorization Headers (8) Body Scripts Settings Cookies

Body (raw JSON)

```

1 {
2   "nombre": "Parque Central",
3   "direccionAproximada": "Calle 10 #20-30",
4   "latitud": 4.65,
5   "longitud": -74.05,
6   "ciudad": {
7     "idCiudad": 1
8   }
9 }

```

201 Created • 94 ms • 206 B • Save Response

Raw Preview Visualize

1 Punto geográfico creado exitosamente

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

HTTP RF9 / RF8 Copy

POST http://localhost:8080/servicios/acciones/solicitar

Params Authorization Headers (8) Body Scripts Settings Cookies

Body (raw JSON)

```

1 {
2   "idUsuario": 251,
3   "tipoServicio": "Transporte Pasajeros",
4   "nivelRequerido": "Estandar",
5   "idPuntoPartida": 10,
6   "destinos": [12, 17],
7   "orden": null,
8   "restaurante": null
9 }
10
11

```

201 Created • 1.85 s • 248 B • Save Response

{} JSON Preview Visualize

```

1 {
2   "mensaje": "Servicio solicitado exitosamente",
3   "servicioId": 1259,
4   "viajeId": 1259
5 }

```

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

HTTP RF9 / RF9

PUT <http://localhost:8080/viajes/{{idViaje}}/finalizar>

Params Authorization Headers (7) Body Scripts Settings Cookies

Body Cookies Headers (5) Test Results ⌂

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body.

200 OK • 598 ms • 1.39 KB ⌂ Save Response ⌂

{ } JSON Preview Visualize ⌂

```
1 {  
2     "mensaje": "Viaje finalizado correctamente",  
3     "viaje": {  
4         "idViaje": 1259,  
5         "fechaHoraInicio": "2025-11-03T20:36:39.555457",  
6         "fechaHoraFin": "2025-11-03T20:36:45.3125473",  
7         "longitudTrayecto": 10.988967953560775,  
8         "idServicio": {  
9             "idServicio": 1259,  
10            "idCliente": {  
11                "idUsuario": 261,  
12                "nombre": "Pasajero_101",  
13                "numeroCelular": "31100000101",  
14                "numeroCedula": "CC000101",  
15                "correoElectronico": "pasajero_101@mail.com",  
16                "tipo": "Cliente"  
17            },  
18            "fechaHora": "2025-11-03T20:36:38.339582",  
19            "tipoServicio": "Transporte Pasajeros",  
20            "nivelRequerido": "Estandar",  
21            "estado": "Finalizado",  
22            "orden": null,  
23            "restaurante": null,  
24            "idPuntoPartida": {  
25                "idPunto": 10,  
26                "nombre": "Punto_10",  
27                "latitud": 4.7,  
28                "longitud": -73.9,  
29                "direccionAproximada": "Calle 10",  
30                "ciudad": {  
31                    "idCiudad": 11,  
32                    "nombre": "Ciudad_11"  
33                }  
34            }  
35        }  
36    }  
37}
```

The screenshot shows the Postman interface with the following details:

- Request Method:** PUT
- Request URL:** <http://localhost:8080/viajes/{{idViaje}}/finalizar>
- Body Content:** This request does not have a body.
- Response Status:** 200 OK
- Response Time:** 598 ms
- Response Size:** 1.39 KB
- Response Headers:** (None listed)
- Response Body:** (JSON format)

```
{  
    "id": 1,  
    "idConductor": 1,  
    "idVehiculo": 1,  
    "idTarifa": 1,  
    "costo": null  
}
```

RF10: Dejar una revisión por parte del usuario de servicios

POST RF10

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

```
1 {
2     "idViaje": 439,
3     "idUsuario": 238,      // Cliente que califica
4     "puntuacion": 4.5,
5     "comentario": "El conductor fue muy amable y puntual"
6 }
7
```

Body Cookies Headers (5) Test Results (1/1)

201 Created 611 ms 1.67 KB Save Response

{ } JSON Preview Visualize

```
1 {
2     "mensaje": "Review creada exitosamente",
3     "review": {
4         "idRevision": 418,
5         "usuarioCalificador": {
6             "idUsuario": 238,
7             "nombre": "Pasajero_88",
8             "numeroCelular": "31100000088",
9             "numeroCedula": "CC0000088",
10            "correoElectronico": "pasajero_88@mail.com",
11            "tipo": "Cliente"
12        },
13        "usuarioCalificado": {
14            "idUsuario": 140,
15            "nombre": "Conductor_140",
16            "numeroCelular": "30000000140",
17            "numeroCedula": "CC000140",
18            "correoElectronico": "conductor_140@mail.com",
19            "tipo": "Conductor"
20        },
21        "viaje": {
22            "idViaje": 439,
23            "fechaHoraInicio": "2025-08-16T00:35:41",
24            "fechaHoraFin": "2025-09-14T13:24:33",
25            "longitudTrayecto": 15.196668529996948,
26            "idServicio": 439
27        }
28    }
29}
```

POST RF10

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

```
1 {
2     "idViaje": 439,
3     "idUsuario": 238,      // Cliente que califica
4     "puntuacion": 4.5,
5     "comentario": "El conductor fue muy amable y puntual"
6 }
7
```

Body Cookies Headers (5) Test Results (1/1)

201 Created 611 ms 1.67 KB Save Response

{ } JSON Preview Visualize

```
1 {
2     "idServicio": 439,
3     "idCliente": {
4         "idUsuario": 238,
5         "nombre": "Pasajero_88",
6         "numeroCelular": "31100000088",
7         "numeroCedula": "CC0000088",
8         "correoElectronico": "pasajero_88@mail.com",
9         "tipo": "Cliente"
10    },
11    "fechaHora": "2025-10-17T01:52:10",
12    "tipoServicio": "Transporte Pasajeros",
13    "nivelRequerido": "Estandar",
14    "estado": "Pendiente",
15    "orden": null,
16    "restaurante": null,
17    "idPuntoPartida": {
18        "idPunto": 39,
19        "nombre": "Punto_39",
20        "latitud": 4.99,
21        "longitud": -73.61,
22        "direccionAproximada": "Calle 39",
23        "ciudad": {
24            "idCiudad": 20,
25            "nombre": "Ciudad_20"
26        }
27    }
28}
```

Find and replace Console Runner Start Proxy Cookies Vault Trash

POST RF10

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

```

1  {
2    "idViaje": 439,
3    "idUser": 238,      // Cliente que califica
4    "puntuacion": 4.5,
5    "comentario": "El conductor fue muy amable y puntual"
6  }

```

Body Cookies Headers (5) Test Results (1/1) 201 Created 611 ms 1.67 KB Save Response

{ } JSON Preview Visualize

```

52  },
53  },
54  "idConductor": {
55    "idUser": 140,
56    "nombre": "Conductor_140",
57    "numeroCelular": "30000000140",
58    "numeroCedula": "CC0000140",
59    "correoElectronico": "conductor_140@mail.com",
60    "tipo": "Conductor"
61  },
62  "idVehiculo": {
63    "idVehiculo": 140,
64    "tipo": "Carro",
65    "marca": "Marca_0",
66    "modelo": "Modelo_140",
67    "color": "Color_0",
68    "placa": "PLA0140",
69    "ciudadExpedicion": "Ciudad_0",
70    "capacidadPasajeros": 6
71  },
72  },
73  "puntuacion": 4.5,
74  "comentario": "El conductor fue muy amable y puntual",
75  "fecha": "2025-11-03T15:11:19.470001"
76}
77

```

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

RF11: Dejar una revisión por parte de un usuario conductor

HTTP RF11 / RF11

POST http://localhost:8080/reviews/registrar

Params Authorization Headers (8) Body Scripts Settings

None form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2     "idViaje": 439,
3     "idUsuario": 140,      // Conductor que califica
4     "puntuacion": 5.0,
5     "comentario": "El pasajero fue muy cordial y puntual"
6 }
```

Body Cookies Headers (5) Test Results (1/1)

201 Created 617 ms 1.67 KB Save Response

```
1 {
2     "mensaje": "Review creada exitosamente",
3     "review": {
4         "idRevision": 419,
5         "usuarioCalificador": {
6             "idUsuario": 140,
7             "nombre": "Conductor_140",
8             "numeroCelular": "380000000140",
9             "numeroCedula": "CC000140",
10            "correoElectronico": "conductor_140@mail.com",
11            "tipo": "Conductor"
12        },
13        "usuarioCalificado": {
14            "idUsuario": 238,
15            "nombre": "Pasajero_88",
16            "numeroCelular": "31100000088",
17            "numeroCedula": "CC000088",
18            "correoElectronico": "pasajero_88@mail.com",
19            "tipo": "Cliente"
20        },
21        "viaje": {
22            "idViaje": 439,
23            "fechaHoraInicio": "2025-08-16T08:35:41",
24            "fechaHoraFin": "2025-09-14T13:24:33",
25            "longitudTrayecto": 15.196668529996948,
26            "idServicio": {
27                "id": ...
28            }
29        }
30    }
31 }
```

HTTP RF11 / RF11

POST http://localhost:8080/reviews/registrar

Params Authorization Headers (8) Body Scripts Settings

None form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2     "idViaje": 439,
3     "idUsuario": 140,      // Conductor que califica
4     "puntuacion": 5.0,
5     "comentario": "El pasajero fue muy cordial y puntual"
6 }
```

Body Cookies Headers (5) Test Results (1/1)

201 Created 617 ms 1.67 KB Save Response

```
1 {
2     "idServicio": {
3         "idServicio": 439,
4         "idCliente": {
5             "idUsuario": 238,
6             "nombre": "Pasajero_88",
7             "numeroCelular": "31100000088",
8             "numeroCedula": "CC000088",
9             "correoElectronico": "pasajero_88@mail.com",
10            "tipo": "Cliente"
11        },
12        "fechaHora": "2025-10-17T01:52:19",
13        "tipoServicio": "Transporte Pasajeros",
14        "nivelRequerido": "Estandar",
15        "estado": "Pendiente",
16        "orden": null,
17        "restaurante": null,
18        "idPuntoPartida": {
19            "idPunto": 39,
20            "nombre": "Punto_39",
21            "latitud": 4.99,
22            "longitud": -73.61,
23            "direccionAproximada": "Calle 39",
24            "ciudad": {
25                "idCiudad": 26,
26                "nombre": "Ciudad_26"
27            }
28        }
29    }
30 }
```

The screenshot shows the Postman interface with the following details:

- Request URL:** `http://localhost:8080/reviews/registrar`
- Method:** POST
- Body Content (Raw JSON):**

```

1 {
2     "idViaje": 439,
3     "idUser": 148,      // Conductor que califica
4     "puntuacion": 5.0,
5     "comentario": "El pasajero fue muy cordial y puntual"
6 }
7
    
```
- Response Status:** 201 Created
- Response Body (Raw JSON):**

```

52         }
53     },
54     "idConductor": {
55         "idUser": 148,
56         "nombre": "Conductor_148",
57         "numeroCelular": "30000000140",
58         "numeroCedula": "CC0000140",
59         "correoElectronico": "conductor_148@mail.com",
60         "tipo": "Conductor"
61     },
62     "idVehiculo": {
63         "idVehiculo": 148,
64         "tipo": "Carro",
65         "marca": "Marca_0",
66         "modelo": "Modelo_140",
67         "color": "Color_0",
68         "placa": "PLA0140",
69         "ciudadExpedicion": "Ciudad_0",
70         "capacidadPasajeros": 6
71     },
72     "puntuacion": 5.0,
73     "comentario": "El pasajero fue muy cordial y puntual",
74     "fecha": "2025-11-03T15:19:54.271Z"
75   }
76 }
77
    
```

RFC1:

Tablas usadas: Usuarios, Servicio, Tarifa y viajes

Atributos usados: Usuarios: idUsuario

Servicios: idServicio, nivelRequerido

Tarifa: tipoServicio, nivel, vigenciaDesde, vigenciaHasta

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;
```

```

1 [
2   {
3     "idServicio": 886,
4     "fecha": "2025-09-19 16:05:56.0",
5     "longitudTrayecto": 16.02,
6     "costoTotal": 5605.75,
7     "tipoServicio": "Transporte Pasajeros",
8     "idVehiculo": 137
9   },
10  {
11    "idServicio": 887,
12    "fecha": "2025-10-10 07:13:30.0",
13    "longitudTrayecto": 10.26,
14    "costoTotal": 3590.39,
15    "tipoServicio": "Transporte Pasajeros",
16    "idVehiculo": 138
17  },
18  {
19    "idServicio": 888,
20    "fecha": "2025-10-02 07:33:51.0",
21    "longitudTrayecto": 10.26,
22    "costoTotal": 3590.39,
23    "tipoServicio": "Transporte Pasajeros",
24    "idVehiculo": 138
25  }
]

```

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;

```

GET <http://localhost:8080/rfc/top-conductores>

Params **Authorization** **Headers (6)** **Body** **Scripts** **Settings** **Cookies**

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body **Cookies** **Headers (5)** **Test Results** **⌚** **200 OK** **89 ms** **1.31 KB** **🔗** **Save Response** **...**

{ } **JSON** **Preview** **Visualize** **⌚**

```

1  [
2    {
3      "idConductor": 1,
4      "nombre": "Conductor_1",
5      "totalViajes": 15
6    },
7    {
8      "idConductor": 25,
9      "nombre": "Conductor_25",
10     "totalViajes": 9
11   },
12   {
13     "idConductor": 23,
14     "nombre": "Conductor_23",
15     "totalViajes": 9
16   },
17   {
18     "idConductor": 21,
19     "nombre": "Conductor_21",
20     "totalViajes": 9
21   },

```

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;

```

The screenshot shows a REST client interface with the following details:

- HTTP:** Proyecto 1 / RFC3
- Method:** GET
- URL:** http://localhost:8080/rfc/ganancias/conductor/1
- Headers:** (6)
- Body:** (JSON) [1]


```

1 [
2   {
3     "idVehiculo": 1,
4     "tipoServicio": "Transporte Pasajeros",
5     "idUsuario": 1,
6     "costoTotal": 44088.0,
7     "ganancia": 35270.0
8   }
9 ]
      
```
- Response Headers:** 200 OK, 47 ms, 274 B
- Test Results:** (1)
- Cookies:** (5)

RFC4:

Tablas usadas: Ciudad, Servicio y PuntoUbicacion

Atributos: Ciudad: nombre, idCiudad

Servicios: tipoServicio, 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;
```

HTTP Proyecto 1 / RFC4

GET http://localhost:8080/rfc/distribucion/servicios?idCiudad=10&fechaInicio=2024-07-04%2023:54:14.000&fechaFin=2025-09-12%2003:41:28.000

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/> idCiudad	10		
<input checked="" type="checkbox"/> fechaInicio	2024-07-04%2023:54:14.000		
<input checked="" type="checkbox"/> fechaFin	2025-09-12%2003:41:28.000		

Body Cookies Headers (5) Test Results | ⏱ 200 OK 47 ms 293 B | Save Response

{ } JSON ▾ Preview Visualize

```

1 [ 
2   {
3     "ciudad": "Ciudad_10",
4     "tipoServicio": "Transporte Pasajeros",
5     "nivelRequerido": "Estandar",
6     "totalServicios": 27,
7     "porcentaje": 100.0
8   }
9 ]

```

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:

HTTP Escenarios RF8 / Crear cliente

POST http://localhost:8080/usuarios/new/cliente

Params Authorization Headers (8) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Schema Beautify

```

1 {
2   "usuario": {
3     "nombre": "Laura SinTarjeta",
4     "correoElectronico": "laura.sintarjeta@uniandes.edu.co",
5     "numeroCelular": "3000000000",
6     "numeroCedula": "CC1016593678",
7     "tipoUsuario": "Cliente"
8   },
9   "tarjeta": null
10 }

```

Body Cookies Headers (5) Test Results ⏱ 201 Created 181 ms 345 B | Save Response

{ } JSON ▾ Preview Visualize

```

1 {
2   "idUsuario": 404,
3   "nombre": "Laura SinTarjeta",
4   "numeroCelular": "3000000000",
5   "numeroCedula": "CC1016593678",
6   "correoElectronico": "laura.sintarjeta@uniandes.edu.co",
7   "tipo": "cliente"
8 }

```

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

HTTP Escenarios RF8 / RF8 Error 1

POST http://localhost:8080/servicios/acciones/solicitar

Body (8) Headers (8) Scripts Settings Cookies

Params Authorization Headers (8) Body (8) Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Schema Beautify

```
1 {
2     "idUsuario": 404,
3     "tipoServicio": "Transporte Pasajeros",
4     "nivelRequerido": "Estandar",
5     "idPuntoPartida": 10,
6     "destinos": [12, 17],
7     "orden": null,
8     "restaurante": null
9 }
10
11
```

Body Cookies Headers (4) Test Results 500 Internal Server Error 196 ms 292 B Save Response

{ } JSON Preview Debug with AI

```
1 {
2     "mensaje": "Error al solicitar el servicio: El usuario no tiene un medio de pago registrado disponible.",
3     "servicioId": null,
4     "viajeId": null
5 }
```

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:

HTTP Escenarios RF8 / Crear tarjeta

POST http://localhost:8080/tarjetas-credito/new/save

Body (8) Headers (8) Scripts Settings Cookies

Params Authorization Headers (8) Body (8) Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Schema Beautify

```
1 {
2     "titularDeLaTarjeta": "Laura Martinez",
3     "numeroTarjeta": "4111111111111111",
4     "fechaExpiracion": "2028-07-01",
5     "codigoSeguridad": 123,
6     "cliente": {
7         "idUsuario": {{idUsuario}}
8     }
9 }
```

Body Cookies Headers (5) Test Results 201 Created 107 ms 208 B Save Response

Raw Preview Visualize

```
1 Tarjeta de crédito creada exitosamente
```

POST | http://localhost:8080/servicios/acciones/solicitar

Body

```

1  {
2    "idUsuario": {{idUsuario}},
3    "tipoServicio": "Transporte Pasajeros",
4    "nivelRequerido": "Estandar",
5    "idPuntoPartida": 10,
6    "destinos": [12, 17],
7    "orden": null,
8    "restaurante": null
9  }
10
11

```

Body

```

1  {
2    "mensaje": "Servicio solicitado exitosamente",
3    "servicioId": 1270,
4    "viajeId": 1268
5  }

```

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:

POST | http://localhost:8080/servicios/acciones/solicitar

Body

```

1  {
2    "idUsuario": {{idUsuario}},
3    "tipoServicio": "Domicilio Comida",
4    "nivelRequerido": "Estandar",
5    "idPuntoPartida": 10,
6    "destinos": [12, 17],
7    "orden": 20,
8    "restaurante": "El corral"
9  }
10
11

```

Body

```

1  {
2    "mensaje": "Error al solicitar el servicio: No hay conductores disponibles para el servicio solicitado.",
3    "servicioId": null,
4    "viajeId": null
5  }

```

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	2025-11-03T22:05:33.0003812	(null)	1284	1	1	1	(null)
2	1282	2025-11-03T22:05:35.387473	(null)	1285	1	1	1	(null)
3	1283	2025-11-03T22:05:37.959969	(null)	1286	1	1	1	(null)
4	1284	2025-11-03T22:05:40.564663	(null)	1287	1	1	1	(null)
5	1285	2025-11-03T22:05:43.407933	(null)	1288	1	1	1	(null)
6	1286	2025-11-03T22:05:45.765985	(null)	1289	1	1	1	(null)
7	1287	2025-11-03T22:05:48.155566	(null)	1290	1	1	1	(null)
8	1288	2025-11-03T22:05:50.798101	(null)	1291	1	1	1	(null)
9	1289	2025-11-03T22:05:59.526333	(null)	1292	1	1	1	(null)
10	1290	2025-11-03T22:06:02.576801	(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)	1312	3	3	1	(null)
10	1310	2025-11-03T22:24:29.254777	(null)	1313	3	3	1	(null)
11	1311	2025-11-03T22:24:32.190528	(null)	1314	4	4	1	(null)
12	1312	2025-11-03T22:24:36.139941	(null)	1315	4	4	1	(null)
13	1313	2025-11-03T22:24:39.465053	(null)	1316	4	4	1	(null)
14	1314	2025-11-03T22:24:42.744720	(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
16	1316	2025-11-03T22:24:49.502493	(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
18	1318	2025-11-03T22:24:55.354338	2025-11-03T22:25:09.710079	10.988967953550775	1321	6	6	1
19	1319	2025-11-03T22:33:19.058865	(null)	1323	6	6	1	(null)
20	1320	2025-11-03T22:33:24.065962	(null)	1324	6	6	1	(null)
21	1321	2025-11-03T22:33:29.796437	(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
23	1323	2025-11-03T22:33:39.016306	(null)	1327	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:

HTTP Proyecto 1 / RFC1 Serializable

GET http://localhost:8080/rfc/viajes/cliente/serializable/328

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Body		200 OK · 30.05 s · 1.66 KB · Save Response ·	
Cookies			
Headers (5)			
Test Results			
{} JSON	Preview Visualize		

```

1 {
2   "antes": [
3     {
4       "idServicio": 886,
5       "fecha": "2025-09-19 16:05:56.0",
6       "longitudTrayecto": 16.02,
7       "costoTotal": 5605.75,
8       "tipoServicio": "Transporte Pasajeros",
9       "idVehiculo": 137
10    },
11    {
12      "idServicio": 887,
13      "fecha": "2025-10-10 07:13:30.0",
14      "longitudTrayecto": 10.26,
15      "costoTotal": 3590.39,
16      "tipoServicio": "Transporte Pasajeros",

```

Read Committed:

GET http://localhost:8080/rfc/viajes/cliente/read-committed/328

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Body		200 OK · 30.04 s · 1.66 KB · Save Response ·	
Cookies			
Headers (5)			
Test Results			
{} JSON	Preview Visualize		

```

1 {
2   "antes": [
3     {
4       "idServicio": 886,
5       "fecha": "2025-09-19 16:05:56.0",
6       "longitudTrayecto": 16.02,
7       "costoTotal": 5605.75,
8       "tipoServicio": "Transporte Pasajeros",
9       "idVehiculo": 137
10    },
11    {
12      "idServicio": 887,
13      "fecha": "2025-10-10 07:13:30.0",
14      "longitudTrayecto": 10.26,
15      "costoTotal": 3590.39,
16      "tipoServicio": "Transporte Pasajeros",

```

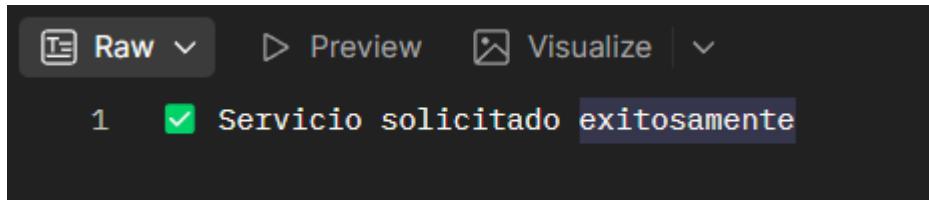
4. Concurrencia Serializable

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 Committed

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.

1 {
 2 "idUsuario": 251,
 3 "tipoServicio": "Transporte Pasajeros",
 4 "nivelRequerido": "Estandar",
 5 "idPuntoPartida": 10,
 6 "destinos": [12],
 7 "orden": null,
 8 "restaurante": null
 9 }
 10
 11

Body Cookies Headers (5) Test Results | 201 Created 873 ms 248 B Save Response

{ } JSON ▾ Preview Visualize

1 {
 2 "mensaje": "Servicio solicitado exitosamente",
 3 "servicioId": 1271,
 4 "viajeId": 1269
 5 }

- 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 COMMITTED, 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 finalizan, sin bloquear ni retrasar la ejecución de la consulta inicial.