Colección Postman - TPI Logística

> Una suite completa de pruebas para validar la arquitectura de microservicios de logística con API Gateway, Keycloak y tres microservicios especializados.

Archivos Incluidos

1. **TPI_Logistica.postman_collection.json**

Colección completa de Postman con:

- **46 requests** organizados en 5 carpetas principales
- **Scripts de test** para encadenamiento automático de variables
- **Bearer Token authentication** para todos los microservicios
- **Ejemplos de body** basados en los DTOs del proyecto

2. **TPI_Logistica.postman_environment.json**

Entorno de Postman con:

- **Variables configurables**: URLs, realms, client IDs
- **Credenciales de usuarios** por rol (CLIENTE, OPERADOR, TRANSPORTISTA)
- **Variables de tokens** que se llenan automáticamente tras autenticación

3. **POSTMAN_GUIA.md**

Guía completa de uso con:

- Instrucciones de instalación paso a paso
- Flujo de ejecución recomendado
- Troubleshooting y solución de errores
- Mapeo de endpoints por rol

4. **POSTMAN_EJEMPLOS.md**

Ejemplos prácticos con:

- 7 escenarios de prueba diferentes
- Pasos detallados para cada caso
- Validaciones de autorización
- Automatización con Collection Runner

Inicio Rápido

1 Descargar los Archivos

```
# Todos están en la raíz del proyecto
ls -la *.postman_*
```

2 Importar en Postman

```
Postman → Import → Seleccionar los 2 archivos JSON → Importar
```

3 Ejecutar Autenticación

```
1. Autenticación > Obtener Token CLIENTE
1. Autenticación > Obtener Token OPERADOR
1. Autenticación > Obtener Token TRANSPORTISTA
```

4 Ejecutar Flujo Completo

```
2. Flujo Principal > Ejecutar en orden (6 requests)
```

🏗️ Estructura de la Colección

```
TPI Logística
├── 1. Autenticación (Keycloak) ..................... 3 requests
│   ├── Obtener Token CLIENTE
│   ├── Obtener Token OPERADOR
│   └── Obtener Token TRANSPORTISTA
│
├── 2. Flujo Principal (Demo) ....................... 6 requests
│   ├── (CLIENTE) Crear Solicitud
│   ├── (OPERADOR) Crear Ruta
│   ├── (OPERADOR) Asignar Camión a Tramo
│   ├── (TRANSPORTISTA) Iniciar Tramo
│   ├── (TRANSPORTISTA) Finalizar Tramo
│   └── (OPERADOR) Finalizar Solicitud
│
├── 3. CRUD - ms-solicitudes ........................ 7 requests
│   ├── Crear Solicitud
│   ├── Listar Solicitudes
│   ├── Obtener Solicitud por ID
│   ├── Obtener Estado de Solicitud
│   ├── Actualizar Estado Contenedor
│   ├── Crear Cliente
│   └── Obtener Cliente por ID
```

```
│
├── 4. CRUD - ms-rutas ............................... 12 requests
│    ├── Crear Ruta
│    ├── Listar Rutas (Paginado)
│    ├── Obtener Rutas por Solicitud
│    ├── Obtener Ruta por ID
│    ├── Asignar Ruta
│    ├── Obtener Costo de Traslado Real
│    ├── Obtener Tarifas Vigentes
│    ├── Asignar Camión a Tramo
│    ├── Iniciar Tramo
│    ├── Finalizar Tramo
│    ├── Listar Tramos por Patente
│    └── Obtener Tramos por Patente (públicos)
│
└── 5. CRUD - ms-camiones ............................ 4 requests
     ├── Crear Camión
     ├── Listar Camiones
     ├── Obtener Camiones Aptos
     └── Obtener Tramos del Transportista
```

🔐 Autenticación

Sistema de Tokens

Todos los requests usan **Bearer Token** con JWT obtenido de Keycloak:

```
Authorization: Bearer {{token_operador}}
```

Flujo de Autenticación

```
1. Request → POST /realms/tpi-backend/protocol/openid-connect/token
2. Parámetros: grant_type=password, username, password, client_id
3. Respuesta: {"access_token": "eyJhbG...", "token_type": "Bearer", ...}
4. Script: pm.environment.set("token_operador", response.access_token)
5. Resultado: {{token_operador}} disponible para todos los requests
```

Credenciales por Rol

Rol	Usuario	Contraseña	Variable
CLIENTE	cliente@mail.com	1234	`{{token_cliente}}`
OPERADOR	operador@mail.com	1234	`{{token_operador}}`
TRANSPORTISTA	transportista@mail.com	1234	`{{token_transportista}}`

Encadenamiento de Variables (Chaining)

Captura Automática de IDs

Los test scripts capturan automáticamente IDs de recursos:

```
// Ejemplo: Crear Solicitud
var jsonData = pm.response.json();
pm.collectionVariables.set("id_solicitud_creada", jsonData.id);
```

Variables de Colección Disponibles

Variable	Origen	Uso
`id_solicitud_creada`	POST /solicitudes	Usado en rutas posteriores
`id_ruta_creada`	POST /rutas	Referencia en operaciones
`id_tramo_1`	POST /rutas (nested)	Asignación de camión

Ejemplo de Encadenamiento

```
// Paso 1: Crear Solicitud
POST {{baseUrl}}/solicitudes
→ Respuesta contiene "id": 42
→ Script guarda: pm.collectionVariables.set("id_solicitud_creada", 42)

// Paso 2: Crear Ruta (usa ID anterior)
POST {{baseUrl}}/rutas
Body:
{
  "idSolicitud": {{id_solicitud_creada}},  ← Usa variable capturada
  "tramos": [...]
}
→ Respuesta contiene "id": 77
→ Script guarda: pm.collectionVariables.set("id_ruta_creada", 77)

// Paso 3: Asignar Camión a Tramo (usa ID del tramo)
PUT {{baseUrl}}/tramos/{{id_tramo_1}}/asignar-camion
```

Variables de Entorno

Configurables (Antes de Usar)

```
{
  "baseUrl": "http://localhost:8080/api",
  "keycloakUrl": "http://localhost:8180",
  "keycloakRealm": "tpi-backend",
  "keycloakClient": "tpi-backend-client",
  "user_cliente": "cliente@mail.com",
  "pass_cliente": "1234",
  "user_operador": "operador@mail.com",
  "pass_operador": "1234",
  "user_transportista": "transportista@mail.com",
  "pass_transportista": "1234"
}
```

Auto-Pobladas (Tras Autenticación)

```
{
  "token_cliente": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_operador": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_transportista": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Casos de Uso Principales

Crear Solicitud de Transporte

```
// CLIENTE crea solicitud
POST {{baseUrl}}/solicitudes
Authorization: Bearer {{token_cliente}}
Content-Type: application/json

{
  "clienteDni": "12345678",
  "pesoContenedor": 500,
  "volumenContenedor": 1000,
  "origenLatitud": -31.4201,
  "origenLongitud": -64.1888,
  "destinoLatitud": -31.5350,
  "destinoLongitud": -64.2637
}
```

Crear Ruta con Tramos

```
// OPERADOR crea ruta
POST {{baseUrl}}/rutas
```

```
Authorization: Bearer {{token_operador}}
Content-Type: application/json

{
  "idSolicitud": {{id_solicitud_creada}},
  "tramos": [
    {
      "idDepositoOrigen": 1,
      "idDepositoDestino": 2,
      "idTipoTramo": 1
    }
  ]
}
```

🚚 Asignar Camión a Tramo

```
// OPERADOR asigna camión
PUT {{baseUrl}}/tramos/{{id_tramo_1}}/asignar-camion
Authorization: Bearer {{token_operador}}
Content-Type: application/json

{
  "patenteCamion": "ABC123",
  "pesoContenedor": 500,
  "volumenContenedor": 1000
}
```

▶️ Iniciar Tramo

```
// TRANSPORTISTA inicia tramo
POST {{baseUrl}}/tramos/{{id_tramo_1}}/iniciar
Authorization: Bearer {{token_transportista}}
Content-Type: application/json

{}
```

⏹️ Finalizar Tramo

```
// TRANSPORTISTA finaliza tramo
POST {{baseUrl}}/tramos/{{id_tramo_1}}/finalizar
Authorization: Bearer {{token_transportista}}
Content-Type: application/json

{}
```

🧪 Pruebas y Validaciones

Test Scripts Incluidos

Cada request de autenticación tiene un test que valida:

```
pm.test("Token obtenido correctamente", function() {
    pm.expect(pm.response.code).to.be.oneOf([200]);
    pm.expect(jsonData.access_token).to.exist;
});
```

Ver Resultados de Tests

1. Ejecuta un request
2. Abre la pestaña **Test Results**
3. Verifica que los tests pasen (verde ✓)

📋 Requisitos Previos

Servicios Corriendo

```
# Verificar que estén en línea
curl http://localhost:8180  # Keycloak
curl http://localhost:8080/api  # API Gateway
curl http://localhost:8081  # ms-solicitudes
curl http://localhost:8082  # ms-rutas
curl http://localhost:8083  # ms-camiones
```

Usuarios en Keycloak

Asegúrate de tener estos usuarios creados con los roles correspondientes:

- `cliente@mail.com` → Rol `CLIENTE`
- `operador@mail.com` → Rol `OPERADOR`
- `transportista@mail.com` → Rol `TRANSPORTISTA`

Docker Compose (si aplica)

```
docker-compose up -d
```

📊 Estadísticas de la Colección

Métrica	Valor
Total de Requests	46
Carpetas	5
Requests de Autenticación	3
Requests del Flujo Principal	6
Requests CRUD	37
Variables de Entorno	13
Variables de Colección	3

🔗 Endpoints por Microservicio

ms-solicitudes

- `POST /solicitudes` - Crear solicitud (CLIENTE)
- `GET /solicitudes` - Listar solicitudes (OPERADOR)
- `GET /solicitudes/{id}` - Obtener solicitud
- `GET /solicitudes/{id}/estado` - Ver estado
- `PUT /solicitudes/{id}/finalizar` - Finalizar (OPERADOR)
- `POST /clientes` - Crear cliente
- `GET /clientes/{id}` - Obtener cliente

ms-rutas

- `POST /rutas` - Crear ruta (OPERADOR)
- `GET /rutas` - Listar rutas (OPERADOR)
- `GET /rutas/{id}` - Obtener ruta
- `GET /rutas/solicitud/{id}` - Rutas por solicitud
- `PUT /tramos/{id}/asignar-camion` - Asignar camión (OPERADOR)
- `POST /tramos/{id}/iniciar` - Iniciar (TRANSPORTISTA)
- `POST /tramos/{id}/finalizar` - Finalizar (TRANSPORTISTA)
- `GET /rutas/tarifas` - Tarifas (público)
- `GET /rutas/solicitud/{id}/costo-real` - Costo (público)

ms-camiones

- `POST /camiones` - Crear camión (OPERADOR)
- `GET /camiones` - Listar camiones (OPERADOR)
- `GET /camiones/buscar-apto` - Buscar por peso/volumen
- `GET /camiones/transportistas/me/tramos` - Mis tramos (TRANSPORTISTA)

📞 Soporte

Problemas Comunes

Problema	Solución
Token inválido	Obtén nuevos tokens en Autenticación
404 Not Found	Verifica que la ID existe (ejecuta requests previos)
403 Forbidden	Comprueba que el usuario tiene el rol correcto
500 Error	Revisa los logs del microservicio

Documentación Relacionada

- `POSTMAN_GUIA.md` - Guía detallada de uso
- `POSTMAN_EJEMPLOS.md` - Escenarios de prueba
- `GUIA_PRUEBAS.md` - Pruebas manuales del proyecto

Notas de Versión

v1.0 - 17/11/2025

- ☑ Colección completa con 46 requests
- ☑ Autenticación OAuth2 con Keycloak
- ☑ Encadenamiento automático de variables
- ☑ Test scripts para validación
- ☑ Documentación completa

Licencia

Esta colección es parte del proyecto **TPI Logística** y está disponible para uso educativo y de desarrollo.

Autor: QA Senior - Sistema TPI Logística
Fecha: 17/11/2025
Versión: 1.0