Ejercicio Real de Práctica con Spring WebFlux y Microservicios Reactivos:

Sistema de Gestión de Transporte Público

Objetivo:

Desarrollar un sistema de microservicios para gestionar y monitorear un sistema de transporte público utilizando Spring WebFlux. El sistema debe ser capaz de manejar grandes volúmenes de datos en tiempo real y proporcionar información actualizada a los usuarios y operadores del sistema.

Descripción del Sistema

El sistema consta de varios microservicios que interactúan entre sí para gestionar la información de rutas, vehículos y pasajeros, y proporcionar actualizaciones en tiempo real sobre la ubicación y el estado de los vehículos.

Requisitos:

1. Microservicio de Gestión de Rutas:

- Descripción: Este servicio gestionará la información sobre las rutas de transporte público, permitiendo operaciones CRUD (Crear, Leer, Actualizar, Eliminar).

- Endpoints:

  - `POST /routes`: Crear una nueva ruta.

  - `GET /routes/{id}`: Obtener detalles de una ruta por ID.

  - `PUT /routes/{id}`: Actualizar información de una ruta.

  - `DELETE /routes/{id}`: Eliminar una ruta.

- Datos: Los datos de ruta deben incluir ID, nombre de la ruta, lista de paradas, y horarios.

2. Microservicio de Gestión de Vehículos:

- Descripción: Este servicio gestionará la información sobre los vehículos, permitiendo operaciones CRUD.

- Endpoints:

  - `POST /vehicles`: Registrar un nuevo vehículo.

  - `GET /vehicles/{id}`: Obtener detalles de un vehículo por ID.

  - `PUT /vehicles/{id}`: Actualizar información de un vehículo.

- `DELETE /vehicles/{id}`: Eliminar un vehículo.

- Datos: Los datos de vehículo deben incluir ID, número de placa, capacidad, y estado actual (en servicio, en mantenimiento, etc.).

3. Microservicio de Monitoreo en Tiempo Real:

- Descripción: Este servicio recibirá y gestionará datos de ubicación en tiempo real de los vehículos.

- Endpoints:

  - `POST /location`: Recibir actualizaciones de ubicación de los vehículos.

  - `GET /location/{vehicleId}`: Obtener la ubicación actual de un vehículo.

- Datos: Los datos de ubicación deben incluir ID del vehículo, latitud, longitud, y timestamp.

4. Microservicio de Gestión de Pasajeros:

- Descripción: Este servicio gestionará la información sobre los pasajeros y sus registros de viaje.

- Endpoints:

  - `POST /passengers`: Registrar un nuevo pasajero.

  - `GET /passengers/{id}`: Obtener detalles de un pasajero por ID.

  - `POST /passengers/{id}/trips`: Registrar un nuevo viaje para un pasajero.

  - `GET /passengers/{id}/trips`: Obtener el historial de viajes de un pasajero.

- Datos: Los datos de pasajero deben incluir ID, nombre, y método de pago preferido.

5. Microservicio de Notificaciones:

- Descripción: Este servicio enviará notificaciones basadas en ciertos eventos o umbrales alcanzados en el sistema.

- Endpoints:

  - `POST /notifications`: Configurar nuevas reglas de notificación.

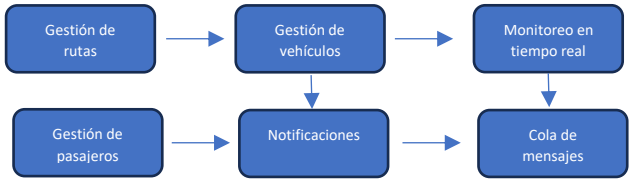
- Funciones: Enviar notificaciones por email o a una cola de mensajes cuando se detecten eventos como retrasos significativos o interrupciones en el servicio.

Infraestructura de Comunicación:

- Descripción: Utilizar una cola de mensajes (por ejemplo, RabbitMQ o Kafka) para la comunicación entre los microservicios.

- Ejemplo: El Microservicio de Monitoreo en Tiempo Real enviará mensajes a la cola cuando reciba actualizaciones de ubicación. El Microservicio de Gestión de Pasajeros puede consumir estos mensajes para actualizar el estado de los viajes en tiempo real.

Diagrama de Servicios para el Sistema de Transporte Público



1. **Gestión de Rutas**: Gestiona la información de las rutas.
2. **Gestión de Vehículos**: Gestiona la información de los vehículos.
3. **Monitoreo en Tiempo Real**: Recibe y procesa los datos de ubicación en tiempo real de los vehículos.
4. **Gestión de Pasajeros**: Gestiona la información de los pasajeros y sus registros de viaje.
5. **Notificaciones**: Envía notificaciones basadas en eventos del sistema.
6. **Cola de Mensajes**: Facilita la comunicación entre los microservicios mediante el uso de una cola de mensajes (por ejemplo, RabbitMQ o Kafka).

Interacciones:

- Los datos de ubicación se envían desde el servicio de **Monitoreo en Tiempo Real** a otros servicios que los necesiten.
- Las notificaciones se generan y envían a partir de eventos capturados por el sistema.
- Los registros de los vehículos y la información de los pasajeros se gestionan entre los servicios correspondientes.

Desafíos Adicionales:

- Escalabilidad: Implementar estrategias de escalabilidad horizontal para manejar incrementos en el volumen de datos.

- Resiliencia: Asegurar que el sistema sea resiliente ante fallos, utilizando patrones como Circuit Breaker y Retry.

- Monitorización: Implementar monitorización y alertas para rastrear el rendimiento y el estado de los microservicios.

- Pruebas: Crear tests de carga para asegurar que el sistema puede manejar grandes volúmenes de datos sin degradación significativa del rendimiento.

Recursos y Herramientas:

- Base de Datos Reactiva: MongoDB, Couchbase

- Cola de Mensajes: RabbitMQ, Kafka

- Monitorización: Prometheus, Grafana

- Framework: Spring WebFlux, Reactor

Este ejercicio te permitirá practicar con Spring WebFlux en un contexto real de gestión de transporte público, mejorando tus habilidades en el manejo de sistemas distribuidos y la programación reactiva.

Ejemplos:

Ejemplo de Ruta en JSON

Aquí tienes un ejemplo de una representación en JSON de una ruta de transporte público. Este ejemplo incluye la información básica de una ruta, como el ID, nombre de la ruta, lista de paradas, y horarios:


```
{
  "routeId": "123",
  "routeName": "Ruta Centro-Norte",
  "stops": [
    {
      "stopId": "1",
      "stopName": "Estación Central",
      "coordinates": {
        "latitude": 40.712776,
        "longitude": -74.005974
      },
      "arrivalTimes": [
        "08:00",
        "08:30",
        "09:00"
      ]
    },
    {
      "stopId": "2",
      "stopName": "Plaza Norte",
      "coordinates": {
        "latitude": 40.730610,
        "longitude": -73.935242
      },
      "arrivalTimes": [
        "08:15",
        "08:45",
        "09:15"
      ]
    },
    {
      "stopId": "3",
      "stopName": "Terminal Norte",
      "coordinates": {
        "latitude": 40.748817,
        "longitude": -73.985428
      },
      "arrivalTimes": [
        "08:30",
        "09:00",
        "09:30"
      ]
    }
  ],
  "schedule": {
    "weekdays": {
      "startTime": "06:00",
      "endTime": "22:00",
```

```
      "frequencyMinutes": 15
    },
    "weekends": {
      "startTime": "07:00",
      "endTime": "20:00",
      "frequencyMinutes": 20
    }
  }
}
```

Explicación del JSON:

1. routeId: Identificador único de la ruta.

2. routeName: Nombre descriptivo de la ruta.

3. stops: Lista de paradas en la ruta, cada una con:

   - stopId: Identificador único de la parada.

   - stopName: Nombre de la parada.

   - coordinates: Coordenadas geográficas de la parada (latitud y longitud).

   - arrivalTimes: Horarios de llegada del vehículo a la parada.

4. schedule: Horarios de operación de la ruta, diferenciados entre días de semana y fines de semana:

   - weekdays: Horarios y frecuencia de operación durante los días de semana.

   - weekends: Horarios y frecuencia de operación durante los fines de semana.

Este ejemplo puede ser adaptado según los requisitos específicos de tu sistema de gestión de transporte público, incluyendo más detalles como tipos de vehículos, información adicional de las paradas, y otros atributos relevantes.

Ejemplo de Vehículo en JSON

Aquí tienes un ejemplo de una representación en JSON de un vehículo utilizado en un sistema de transporte público. Este ejemplo incluye información básica del vehículo, como su ID, número de placa, capacidad, y estado actual.


```
{
  "vehicleId": "V-456",
  "licensePlate": "ABC-1234",
  "capacity": 40,
  "currentStatus": "in_service",
  "type": "bus",
  "routeId": "123",
  "lastMaintenance": "2024-06-15",
  "currentLocation": {
    "latitude": 40.730610,
    "longitude": -73.935242
  },
  "driver": {
    "driverId": "D-789",
    "name": "John Doe",
    "contact": "+1234567890"
  }
}
```


Explicación del JSON:

1. vehicleId: Identificador único del vehículo.

2. licensePlate: Número de placa del vehículo.

3. capacity: Capacidad de pasajeros que puede transportar el vehículo.

4. currentStatus: Estado actual del vehículo, que puede ser "in_service", "out_of_service", "maintenance", etc.

5. type: Tipo de vehículo, por ejemplo, "bus", "tram", "train".

6. routeId: Identificador de la ruta que actualmente está asignada al vehículo.

7. lastMaintenance: Fecha de la última vez que el vehículo recibió mantenimiento.

8. currentLocation: Ubicación actual del vehículo con coordenadas de latitud y longitud.

9. driver: Información del conductor asignado al vehículo, incluyendo:

   - driverId: Identificador único del conductor.

   - name: Nombre del conductor.

   - contact: Información de contacto del conductor.

Ejemplo JSON de un vehículo en mantenimiento


```
{
  "vehicleId": "V-789",
  "licensePlate": "XYZ-5678",
  "capacity": 50,
  "currentStatus": "maintenance",
  "type": "tram",
  "lastMaintenance": "2024-07-01",
  "maintenanceDetails": {
    "scheduledBy": "Jane Smith",
    "scheduledDate": "2024-07-10",
    "details": "Routine check and brake replacement"
  }
}
```


Explicación adicional del JSON:

1. maintenanceDetails: Información adicional sobre el mantenimiento programado:

   - scheduledBy: Nombre de la persona que programó el mantenimiento.

   - scheduledDate: Fecha programada para el mantenimiento.

   - details: Descripción de las tareas de mantenimiento que se realizarán.

Estos ejemplos pueden ser adaptados y extendidos según los requisitos específicos de tu sistema de gestión de transporte público, incluyendo más detalles como registros de mantenimiento anteriores, información adicional del conductor, y otros atributos relevantes.

Ejemplo de Monitoreo en Tiempo Real en JSON

Aquí tienes un ejemplo de una representación en JSON de los datos de monitoreo en tiempo real para un sistema de transporte público. Este ejemplo incluye la información básica sobre la ubicación y estado de un vehículo en tiempo real.


```
{
  "vehicleId": "V-456",
  "timestamp": "2024-07-04T14:48:00Z",
  "location": {
    "latitude": 40.730610,
    "longitude": -73.935242
  },
  "speed": 45.0,
  "direction": "north",
  "routeId": "123",
  "passengerCount": 30,
  "status": "on_route",
  "events": [
    {
      "eventId": "E-001",
      "type": "stop_arrival",
      "stopId": "2",
      "timestamp": "2024-07-04T14:47:00Z"
    },
    {
      "eventId": "E-002",
      "type": "stop_departure",
      "stopId": "2",
      "timestamp": "2024-07-04T14:48:00Z"
    }
  ]
}
```


Explicación del JSON:

1. vehicleId: Identificador único del vehículo.

2. timestamp: Marca de tiempo en que se registró la información.

3. location: Ubicación actual del vehículo con coordenadas de latitud y longitud.

4. speed: Velocidad actual del vehículo en km/h.

5. direction: Dirección del movimiento del vehículo (norte, sur, este, oeste).

6. routeId: Identificador de la ruta que actualmente está siguiendo el vehículo.

7. passengerCount: Número actual de pasajeros a bordo del vehículo.

8. status: Estado actual del vehículo, por ejemplo, "on_route", "stopped", "off_route".

9. events: Lista de eventos recientes relacionados con el vehículo, cada uno con:

   - eventId: Identificador único del evento.

- type: Tipo de evento, como "stop_arrival" (llegada a una parada), "stop_departure" (salida de una parada).

- stopId: Identificador de la parada relacionada con el evento.

- timestamp: Marca de tiempo en que ocurrió el evento.

Ejemplo JSON de un vehículo con un evento de emergencia

```
{
  "vehicleId": "V-789",
  "timestamp": "2024-07-04T15:00:00Z",
  "location": {
    "latitude": 40.748817,
    "longitude": -73.985428
  },
  "speed": 0.0,
  "direction": "stopped",
  "routeId": "456",
  "passengerCount": 25,
  "status": "emergency",
  "events": [
    {
      "eventId": "E-003",
      "type": "emergency_stop",
      "timestamp": "2024-07-04T15:00:00Z",
      "details": "Engine failure detected, vehicle stopped"
    }
  ]
}
```

Explicación adicional del JSON:

1. status: El estado "emergency" indica que el vehículo está en una situación de emergencia.

2. events:

- type: El tipo de evento "emergency_stop" indica que el vehículo ha detenido su marcha debido a una emergencia.

- details: Descripción del evento de emergencia, en este caso, una falla en el motor.

Estos ejemplos pueden ser adaptados y extendidos según los requisitos específicos de tu sistema de gestión de transporte público, incluyendo más detalles como información de diagnóstico del vehículo, comunicaciones con el centro de control, y otros atributos relevantes.

Ejemplo de Notificación en JSON

Aquí tienes un ejemplo de una notificación en JSON para un sistema de transporte público. Esta notificación podría ser enviada a los usuarios o al personal de operaciones en caso de eventos importantes como retrasos, emergencias, o cambios en el servicio.


```
{
  "notificationId": "N-001",
  "timestamp": "2024-07-04T15:05:00Z",
  "type": "
",
  "title": "Interrupción del Servicio en Ruta 456",
  "message": "El servicio en la Ruta 456 se ha interrumpido debido a una
falla mecánica en uno de los vehículos. Se están tomando medidas para
resolver el problema. Por favor, espere actualizaciones.",
  "severity": "high",
  "routeId": "456",
  "vehicleId": "V-789",
  "recipients": [
    {
      "recipientId": "U-123",
      "recipientType": "passenger",
      "contact": {
        "email": "passenger123@example.com",
        "phone": "+123456789"
      }
    },
    {
      "recipientId": "S-456",
      "recipientType": "staff",
      "contact": {
        "email": "staff456@example.com",
        "phone": "+1987654321"
      }
    }
  ],
  "actions": [
    {
      "actionId": "A-001",
      "type": "send_email",
      "status": "completed"
    },
    {
      "actionId": "A-002",
      "type": "send_sms",
      "status": "completed"
    }
  ]
}
```


Explicación del JSON:

1. notificationId: Identificador único de la notificación.

2. timestamp: Marca de tiempo en que se generó la notificación.

3. type: Tipo de notificación, por ejemplo, "service_disruption" (interrupción del servicio), "emergency", "delay".

4. title: Título de la notificación.

5. message: Mensaje detallado de la notificación.

6. severity: Nivel de severidad de la notificación, por ejemplo, "low", "medium", "high".

7. routeId: Identificador de la ruta afectada.

8. vehicleId: Identificador del vehículo relacionado con el evento.

9. recipients: Lista de destinatarios de la notificación, incluyendo:

   - recipientId: Identificador único del destinatario.

   - recipientType: Tipo de destinatario, por ejemplo, "passenger" (pasajero), "staff" (personal).

   - contact: Información de contacto del destinatario, incluyendo email y teléfono.

10. actions: Lista de acciones realizadas como parte de la notificación, cada una con:

    - actionId: Identificador único de la acción.

    - type: Tipo de acción, por ejemplo, "send_email" (enviar email), "send_sms" (enviar SMS).

    - status: Estado de la acción, por ejemplo, "pending", "completed".

Ejemplo JSON de una notificación de retraso


```
{
  "notificationId": "N-002",
  "timestamp": "2024-07-04T15:10:00Z",
  "type": "delay",
  "title": "Retraso en la Ruta 123",
  "message": "El vehículo en la Ruta 123 está experimentando un retraso de
10 minutos debido a tráfico pesado. Agradecemos su paciencia.",
  "severity": "medium",
  "routeId": "123",
  "vehicleId": "V-456",
  "recipients": [
    {
      "recipientId": "U-124",
      "recipientType": "passenger",
      "contact": {
        "email": "passenger124@example.com",
        "phone": "+123456780"
      }
    }
  ],
  "actions": [
    {
      "actionId": "A-003",
      "type": "send_email",
      "status": "completed"
    }
  ]
}
```


Explicación adicional del JSON:

- type: En este caso, el tipo de notificación es "delay" (retraso).

- message: El mensaje indica un retraso específico y la razón del mismo.

- recipients: Solo hay un destinatario en este ejemplo.

- actions: Una sola acción de envío de email ha sido completada.

Estos ejemplos pueden ser adaptados y extendidos según los requisitos específicos de tu sistema de gestión de transporte público, incluyendo más detalles sobre eventos específicos, formatos de contacto adicionales, y tipos de acciones realizadas.

Ejemplo de Gestión de Pasajeros en JSON


```
{
  "passengerId": "P-001",
  "name": "Juan Pérez",
  "email": "juan.perez@example.com",
  "phone": "+1234567890",
  "preferredPaymentMethod": "card",
  "registeredAt": "2024-01-15T09:00:00Z",
  "trips": [
    {
      "tripId": "T-1001",
      "routeId": "123",
      "vehicleId": "V-456",
      "startTime": "2024-07-04T08:00:00Z",
      "endTime": "2024-07-04T08:45:00Z",
      "startStop": "Estación Central",
      "endStop": "Plaza Norte",
      "fare": 2.50
    },
    {
      "tripId": "T-1002",
      "routeId": "123",
      "vehicleId": "V-456",
      "startTime": "2024-07-04T09:00:00Z",
      "endTime": "2024-07-04T09:45:00Z",
      "startStop": "Plaza Norte",
      "endStop": "Terminal Norte",
      "fare": 2.50
    }
  ]
}
```


Explicación del JSON:

1. **passengerId**: Identificador único del pasajero.
2. **name**: Nombre del pasajero.
3. **email**: Correo electrónico del pasajero.
4. **phone**: Número de teléfono del pasajero.
5. **preferredPaymentMethod**: Método de pago preferido del pasajero, por ejemplo, "card" (tarjeta) o "cash" (efectivo).
6. **registeredAt**: Fecha y hora en que el pasajero se registró en el sistema.
7. **trips**: Lista de viajes realizados por el pasajero, cada uno con:
    - **tripId**: Identificador único del viaje.
    - **routeId**: Identificador de la ruta que siguió el viaje.
    - **vehicleId**: Identificador del vehículo utilizado.
    - **startTime**: Hora de inicio del viaje.
    - **endTime**: Hora de finalización del viaje.
    - **startStop**: Nombre de la parada de inicio.
    - **endStop**: Nombre de la parada final.
    - **fare**: Tarifa pagada por el viaje.

Ejemplo de Registro de un Nuevo Pasajero

```
{
  "passengerId": "P-002",
  "name": "Ana Gómez",
  "email": "ana.gomez@example.com",
  "phone": "+1987654321",
  "preferredPaymentMethod": "cash",
  "registeredAt": "2024-07-01T10:30:00Z"
}
```

Explicación del JSON:

- Este ejemplo muestra cómo se registraría un nuevo pasajero en el sistema, con toda la información básica pero sin viajes registrados aún.

Ejemplo de Registro de un Nuevo Viaje

```
{
  "tripId": "T-1003",
  "routeId": "124",
  "vehicleId": "V-457",
  "passengerId": "P-002",
  "startTime": "2024-07-04T10:00:00Z",
  "endTime": "2024-07-04T10:30:00Z",
  "startStop": "Estación Norte",
  "endStop": "Centro Comercial",
  "fare": 3.00
}
```

Explicación del JSON:

- **tripId**: Identificador único del viaje.
- **routeId**: Identificador de la ruta seguida.
- **vehicleId**: Identificador del vehículo utilizado.
- **passengerId**: Identificador del pasajero que realizó el viaje.
- **startTime**: Hora de inicio del viaje.
- **endTime**: Hora de finalización del viaje.
- **startStop**: Nombre de la parada de inicio.
- **endStop**: Nombre de la parada final.
- **fare**: Tarifa pagada por el viaje.

Estos ejemplos pueden ser adaptados y extendidos según los requisitos específicos de tu sistema de gestión de transporte público, incluyendo más detalles sobre métodos de pago, información adicional de contacto, y otros atributos relevantes.