Taller
Java Enterprise

Informe de
Iteración II
Lógica de negocio

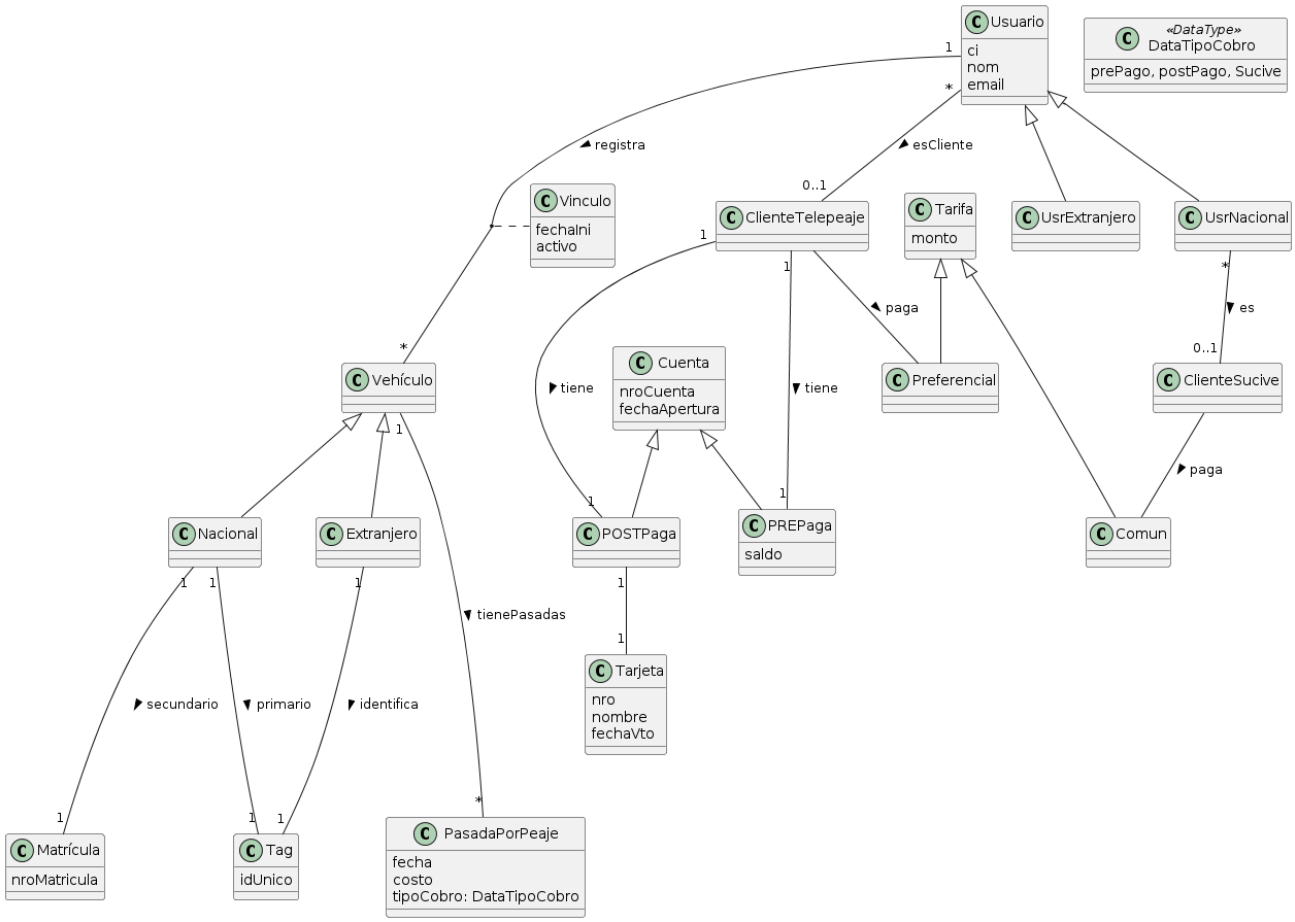
Integrantes:

Carlos Santana	**CI:** 5.017.001-9
Andrés Romano	**CI:** 4.760.801-9
Juan Pilón	**CI:** 4.692.663-6
Nicolás Silva	**CI:** 4.885.055-6

Docente: Gabriel Aramburu

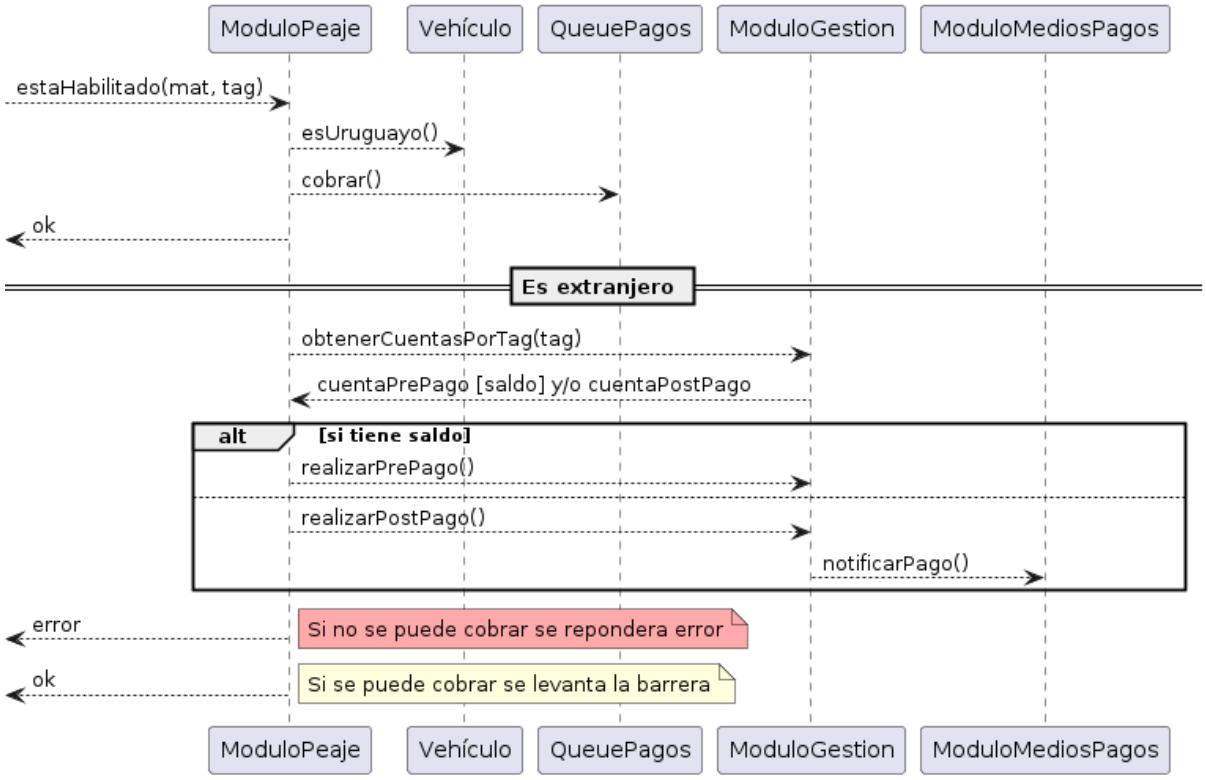
Índice

1 Análisis: modelo de dominio

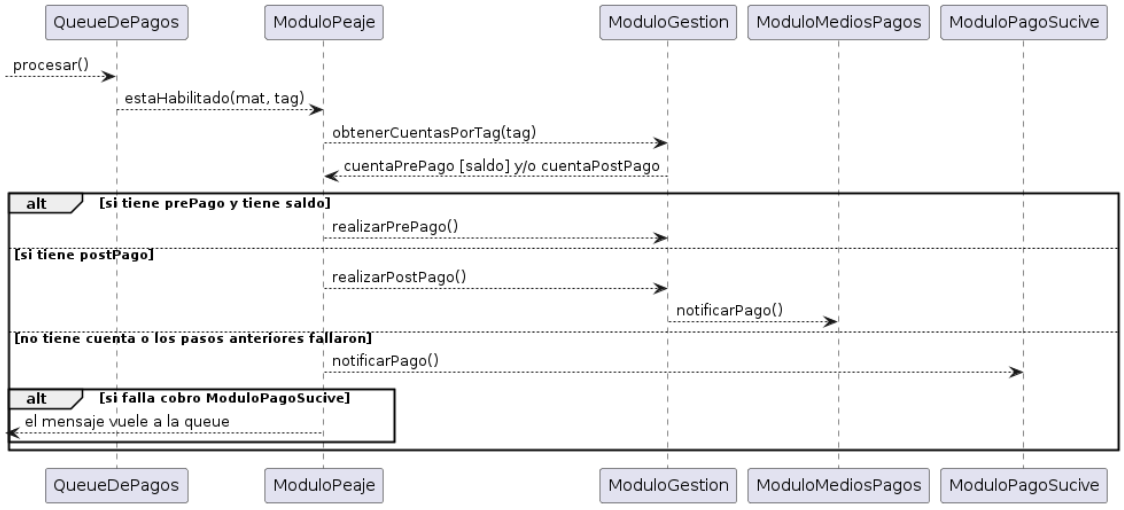


2 Diseño.

2.1 *Casos de uso principal: pasaje de vehículos nacionales*



2.2 *Casos de uso principal*



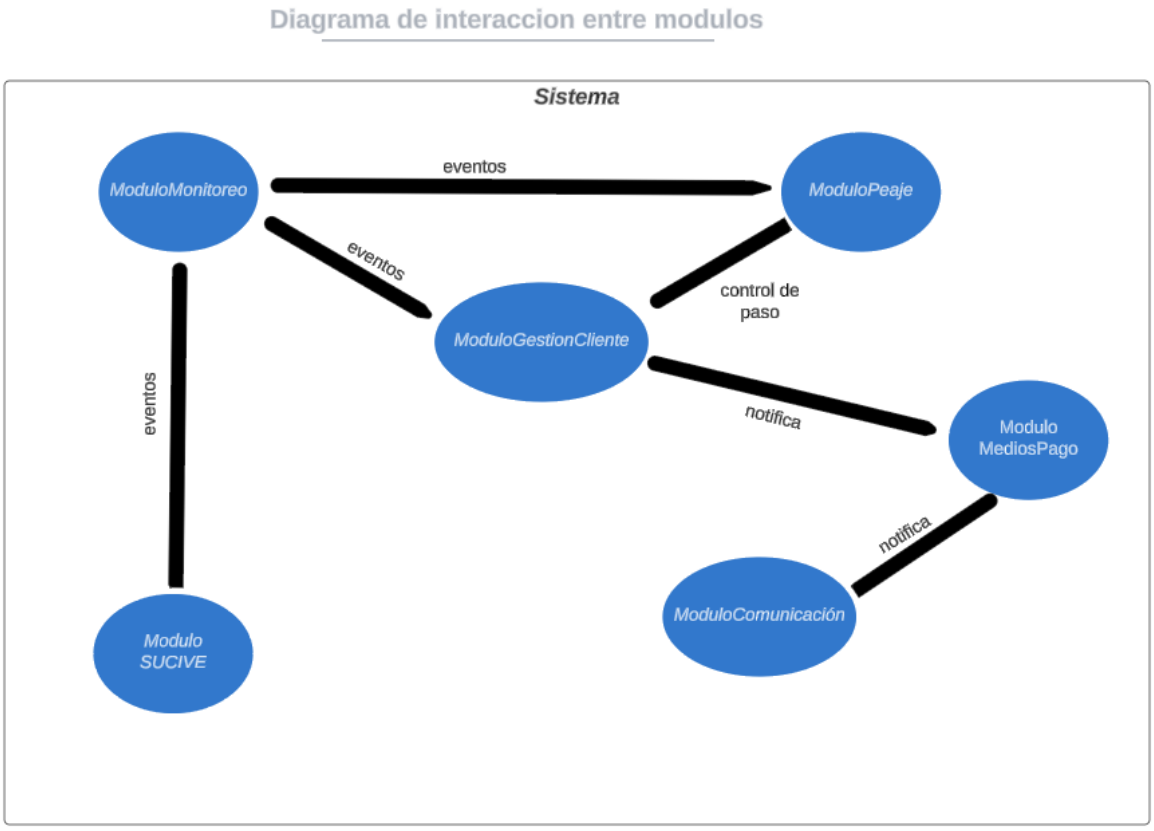
3 Estructura de paquetes.

Estructura

- Módulo de Monitoreo
  - Aplicación
  - Dominio
    - Repo
- Módulo Gestión Clientes
  - Aplicación
  - Dominio
    - Repo
- Módulo Comunicación
  - Aplicación
  - Dominio
    - Repo
- Módulo de Medios Pago
  - Aplicación
  - Dominio
- Módulo de Sucive
  - Aplicación
  - Dominio
    - Repo
- Módulo de Peaje
  - Aplicación
  - Dominio
    - Repo
- Test

4 Desarrollo

4.1 Interacción entre Módulos

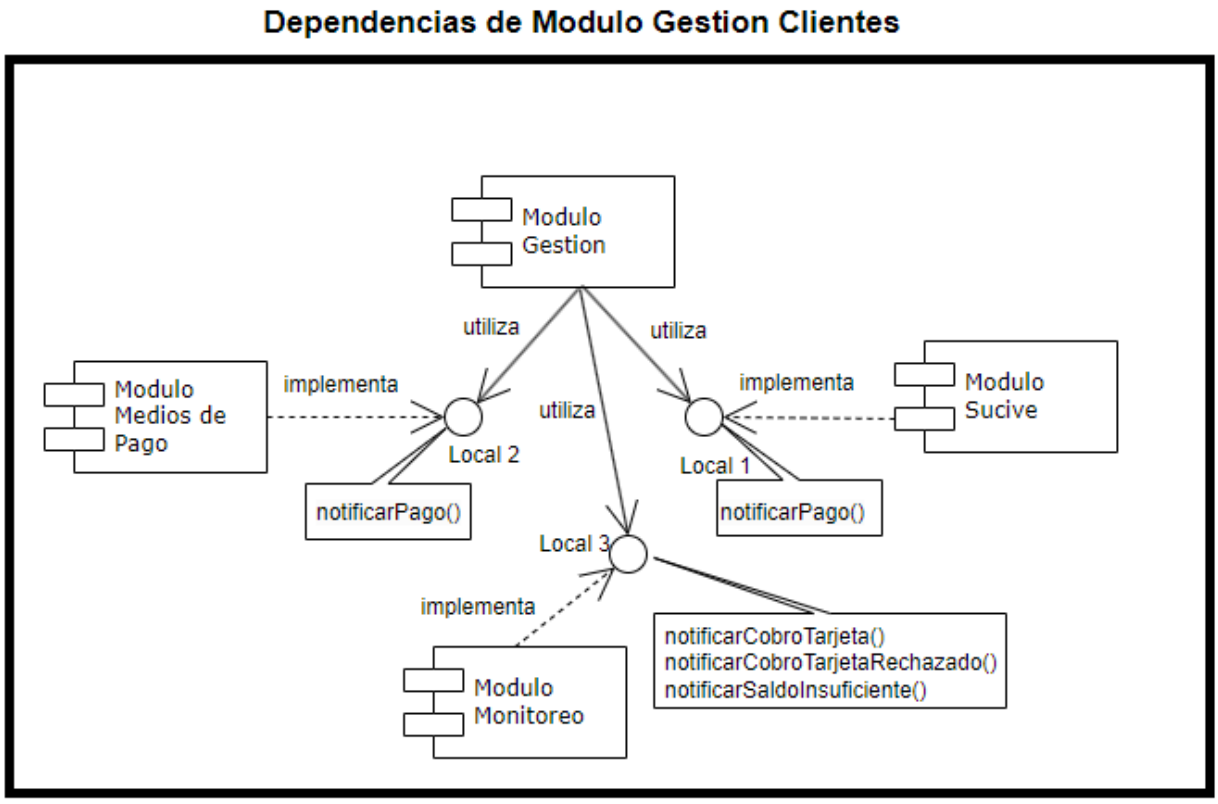


4.1.1 Observaciones

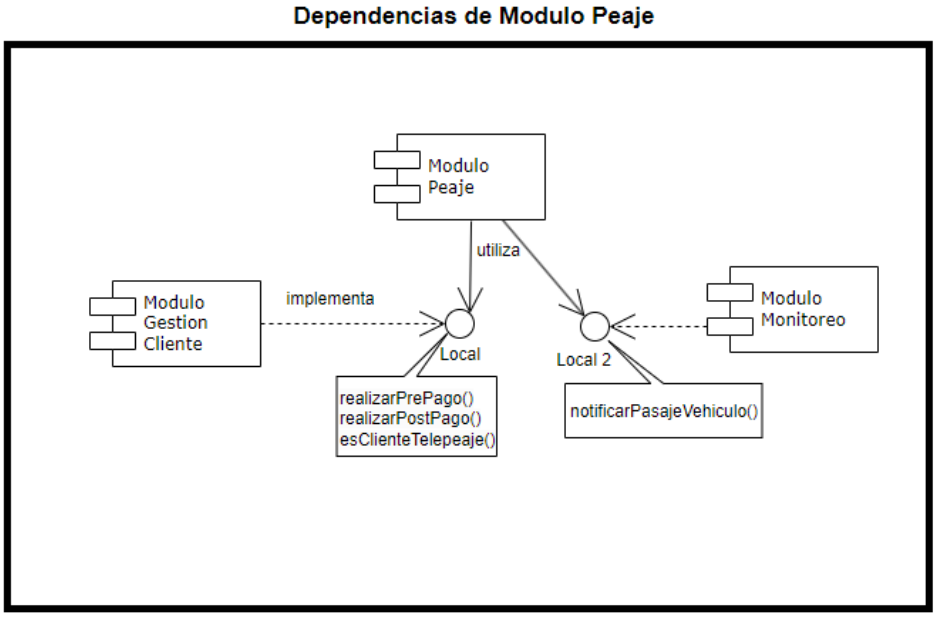
El proyecto se orientó fuertemente a ser escalable y fácil de modificar. En el diagrama se observa una breve interpretación de como los módulos mantienen su comunicación. La mayoría de los mismos operan internamente resolviendo múltiples problemáticas y en los casos que ha sido necesario hasta ahora se han extraído los datos de otros módulos.

4.2 Dependencias entre Módulos

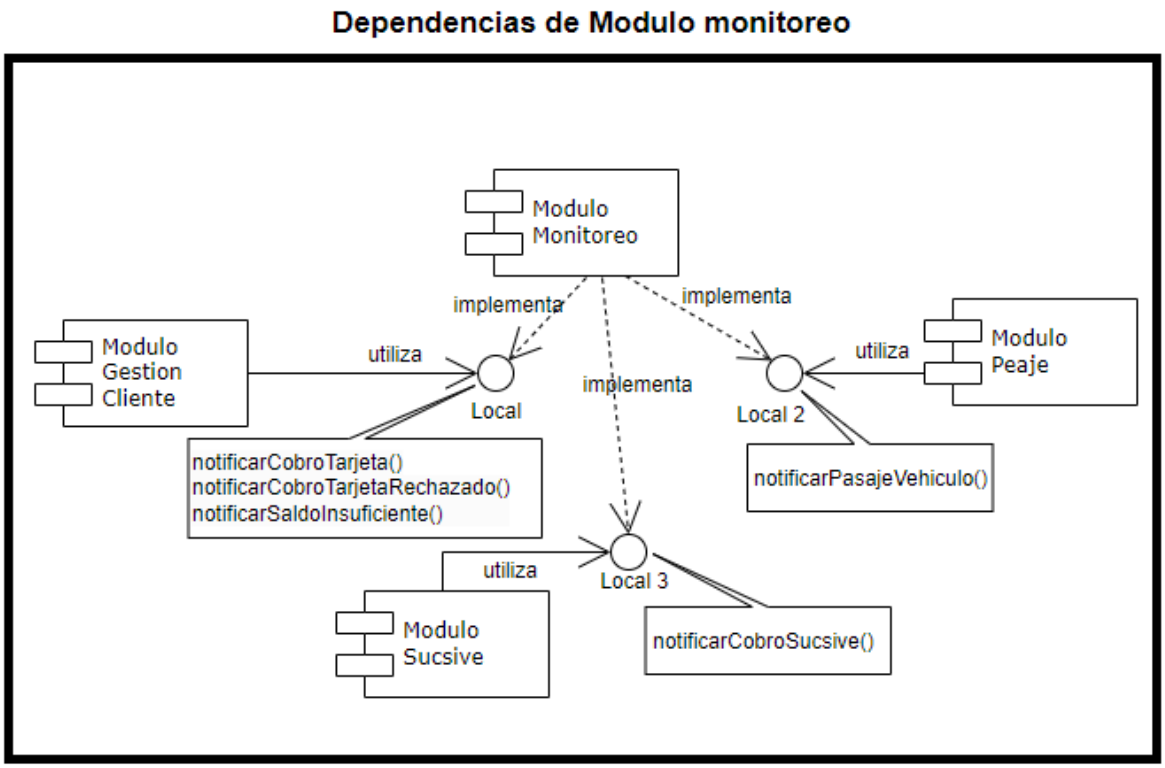
4.2.1 Módulo Gestión de clientes



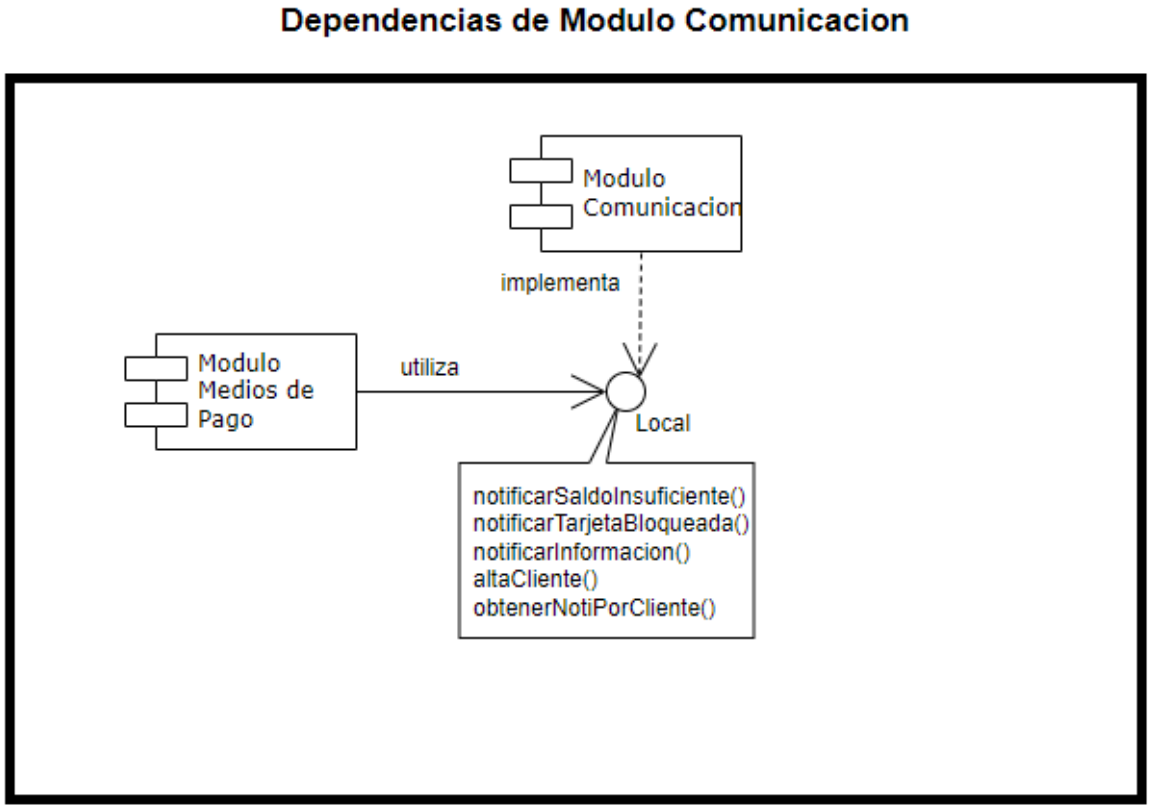
4.2.2 Módulo Peaje



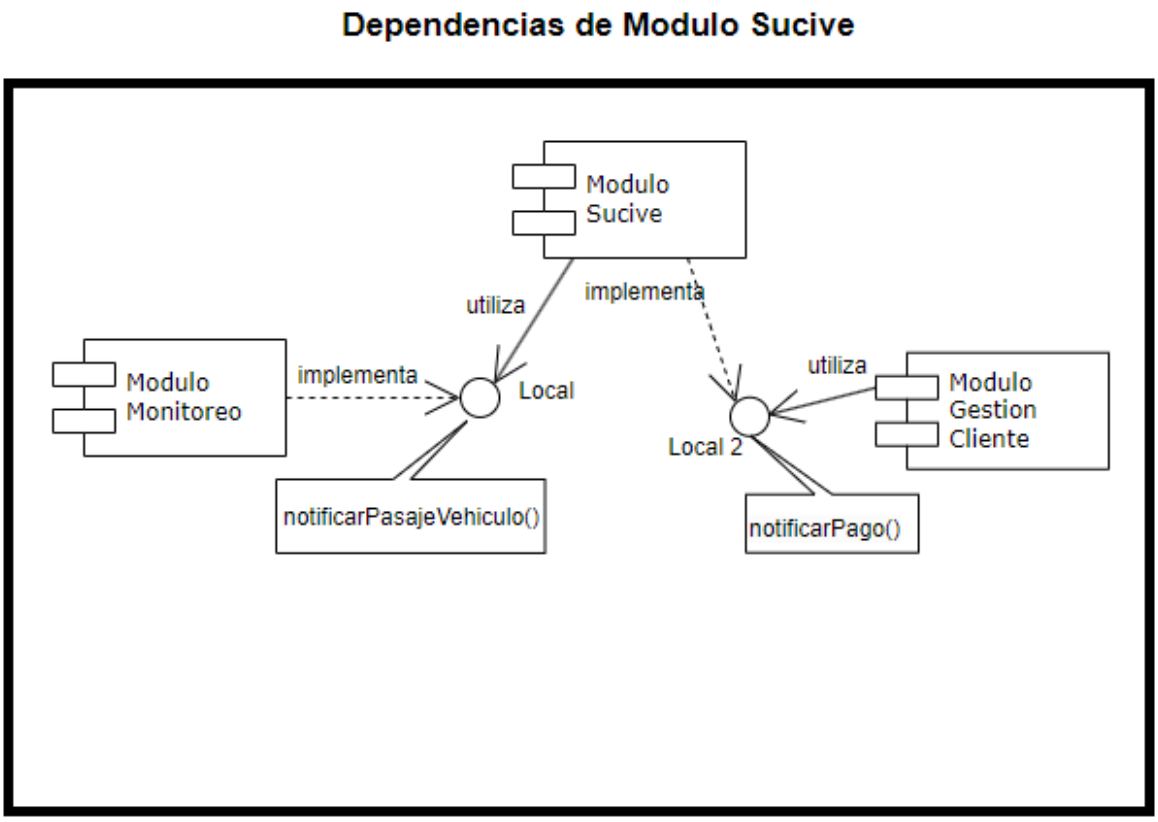
4.2.3 Módulo Monitoreo



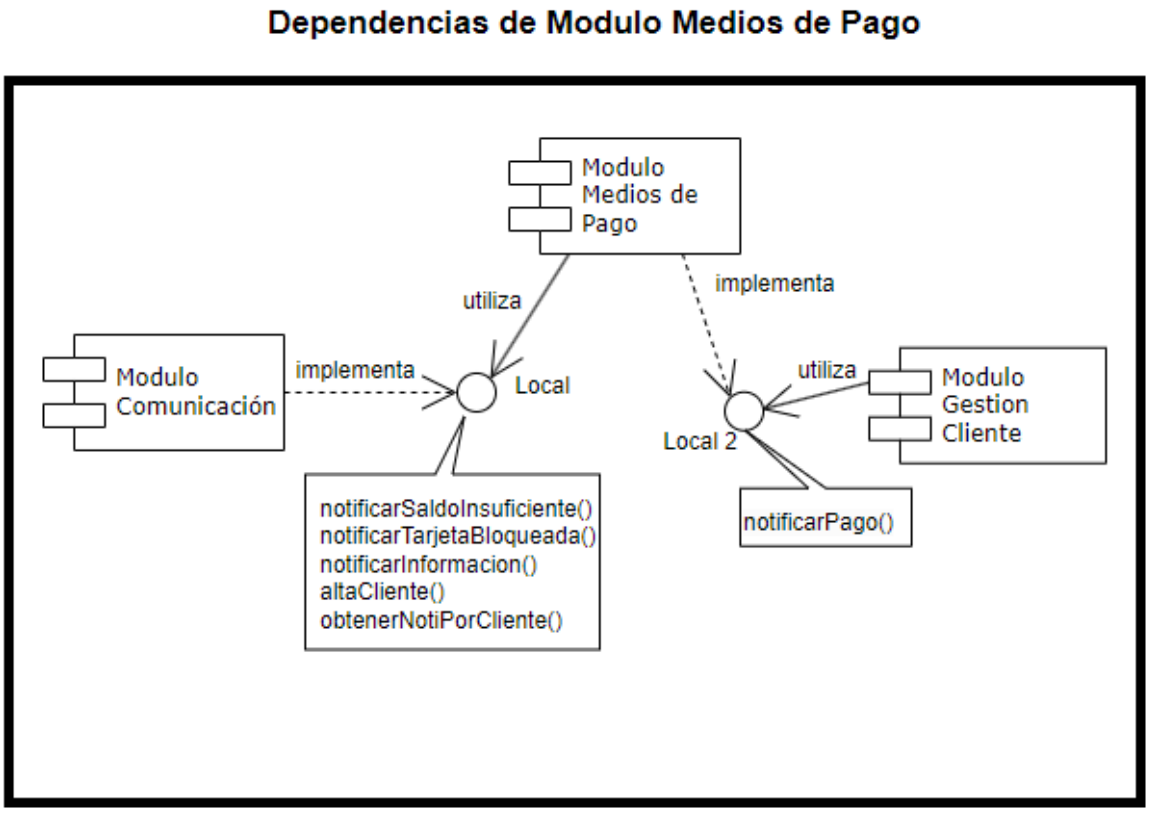
4.2.4 Módulo Comunicación



4.2.5 Módulo Sucive



4.2.6 Módulo Medios de Pago



4.3 Desarrollo de Operaciones

4.3.1 Módulo de Peaje.

estaHabilitado (Identificador): boolean	puede venir tag o matrícula true: habilitado para pasar false: no habilitado
actualizarTarifaComun (importe)	actualiza el costo de la tarifa común
actualizarTarifaPreferencial (preferencial)	actualiza el costo de la tarifa preferencial

Modulo peaje se encarga de gestionar la entrada y salida de los vehículos que pasan por el peaje. Independientemente de si llega un cliente nacional o uno extranjero. En el caso de ser extranjero se deberá procesar el cobro para habilitar su pasaje para ese caso. En el caso nacional se corroborará que la matricula este en sistema.

Luego en este módulo también se manejaran los costos de las tarifas para los cobros en el peaje.

4.3.2 Módulo de gestión de clientes.

altaClienteTelepeaje (Usuario)	registra a un usuario como cliente Telepeaje
vincularVehiculo (Cliente, Vehículo)	asocia el vehículo al Usuario
desvincularVehículo (Cliente, Vehículo)	desvincula a un cliente de un vehículo (por ejemplo en caso de venta)
mostraVehículosVinculados (Cliente):set(Vehículos)	devuelve los vehículos asociados al cliente
cargarSaldo (Cliente, importe)	cargaSaldo a la cuenta PRE paga del cliente
consultarSaldo(Cliente) :importe	devuelve el saldo actual de la cuenta PRE paga del cliente
asociarTarjeta (Cliente, Tarjeta)	asocia tarjeta de crédito a la cuenta POST paga del cliente
consultarPasadas (Cliente, rangoFechas) :set(Pasadas)	devuelve las pasadas realizadas por todos los vehículos registrados en un rango de fechas.
consultarPasadas (Cliente, Vehículo, rangoFechas) :set(Pasadas)	idem anterior, pero solo para un vehículo
obtenerCuentasPorTag(tag): set(cuenta)	devuelve los tipos de cuentas asociadas al cliente Telepeaje Si la cuenta es de PrePago, devuelve el saldo actual **Utilizado por el módulo de Peaje.**
realizarPrePago(importe)	descuenta el importe del pago al saldo del cliente. **Utilizado por el módulo de Peaje.**
realizarPostPAgo(importe)	realiza el pago utilizando tarjeta de crédito

En este módulo se manejó y gestiono la recepción de usuarios al sistema. Se le asociara a un vehículo o se desvinculara en caso de ser necesario. Así como se llevará el registro de la información de cada usuario y las veces que hayan pasado por el peaje.

4.3.3 Módulo de Sucive.

notificarPago(matrícula, importe)	notifica del pago al Sistema externo de Sucive
consultaDePagos(rangoFechas):set(Pagos)	importe total de pagos realizados agrupados por día
consultaDePagos(matrícula):set(Pagos)	devuelve los pagos realizados por sucive para un vehículo en especial

Modulo externo que manejara en una futura iteración el manejo con el api de sucive.

4.3.4 Módulo Medios de Pago.

altaCliente(Cliente, Tarjeta)	Da de alta al cliente
notificarPago(Cliente, Vehículo, importe, Tarjeta)	notifica del pago al Sistema externo de Medios De Pago
consultaDePagos(rangoFechas):set(Pagos)	importe total de pagos realizados agrupados por día
consultaDePagos(Cliente):set(Pagos)	devuelve los pagos realizados por un Cliente en particular
consultaDePagos(Cliente, Vehículo):set(Pagos)	devuelve los pagos realizados por un vehículo determinado de un Cliente en particular

Modulo externo que manejara en una futura iteración el manejo con el api de pago elegida para el manejo del cobro a los clientes.

4.3.5 Módulo de Comunicación

notificarSaldoInsuficiente (Cliente)	notifica vía email al Cliente que su saldo de cuenta PREpaga es insuficiente
notificarTarjetaBloqueada (Cliente)	notifica vía email al Cliente que su tarjeta fue bloqueada
notificarInformacion(texto)	notifica vía email al Cliente alguna información relevante
altaCliente(Cliente)	Da de alta al cliente
obtenerNotiPor-Cliente(Cliente):set(Notificaciones)	devuelve las notificaciones de un cliente en particular

Encargado de en futuras iteraciones enviar mails para notificar a los clientes diversos eventos.

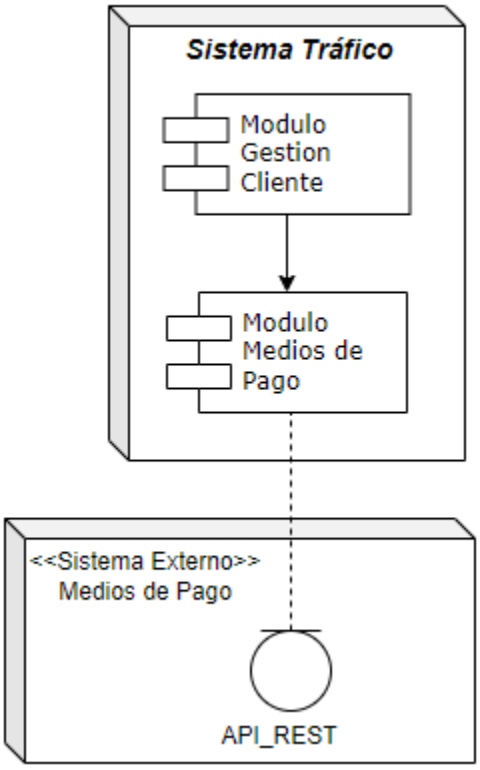
4.3.6 Módulo de Monitoreo

notificarPasajeVehículo()	envía evento que representa el pasaje de un vehículo
notificarCobroSucive()	envía evento que representa el cobro con sucive
notificarCobroTarjeta()	envía evento que representa el cobro con tarjeta
notificarCobroTarjetaRe-chazado()	envía evento que representa el rechazo de cobro por tarjeta
notificarSaldoInsufi-ciente()	envía evento que representa el rechazo de cobro por saldo insuficiente

Es el modulo que guarda y desarrolla las interacciones del sistema con el exterior del mismo. Se encarga de disparar eventos cuando sean necesarios.

5 Api Restfull

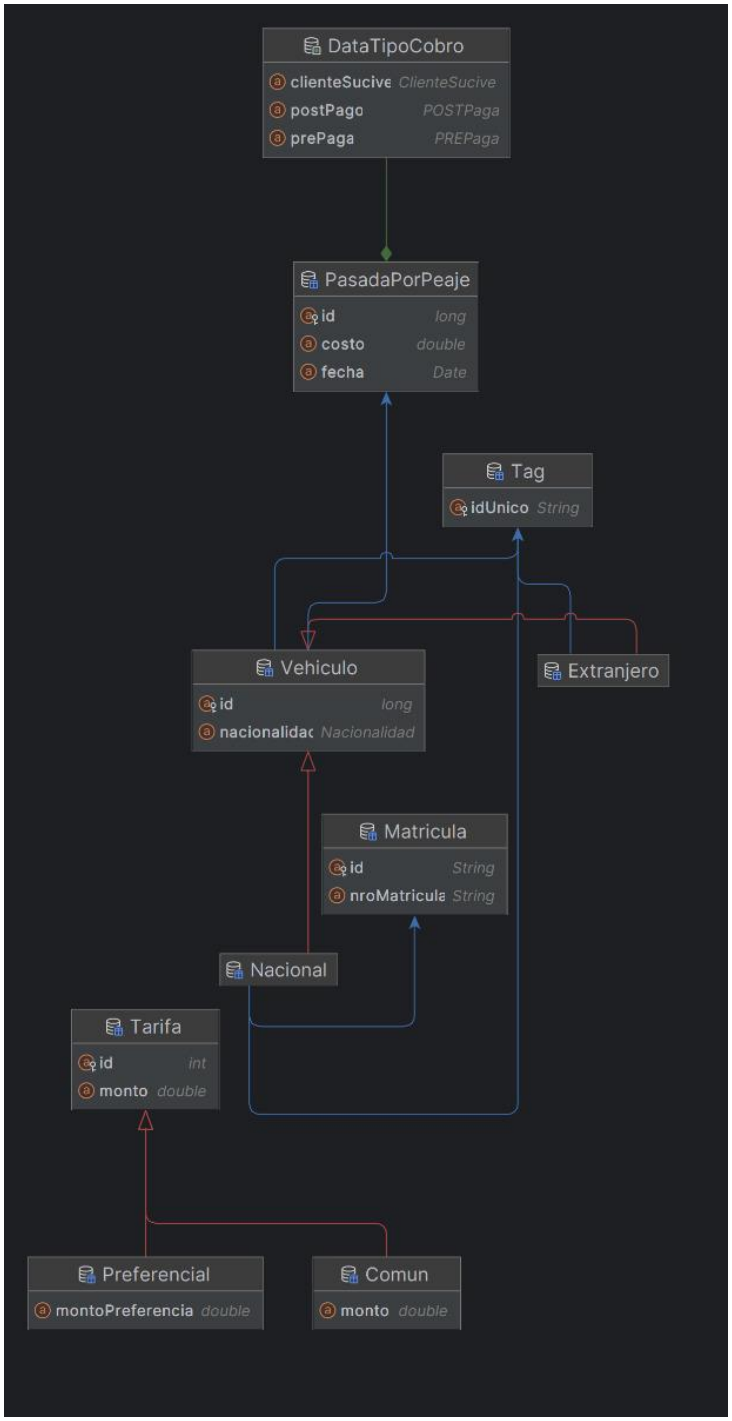
5.1 Conexiones implementadas



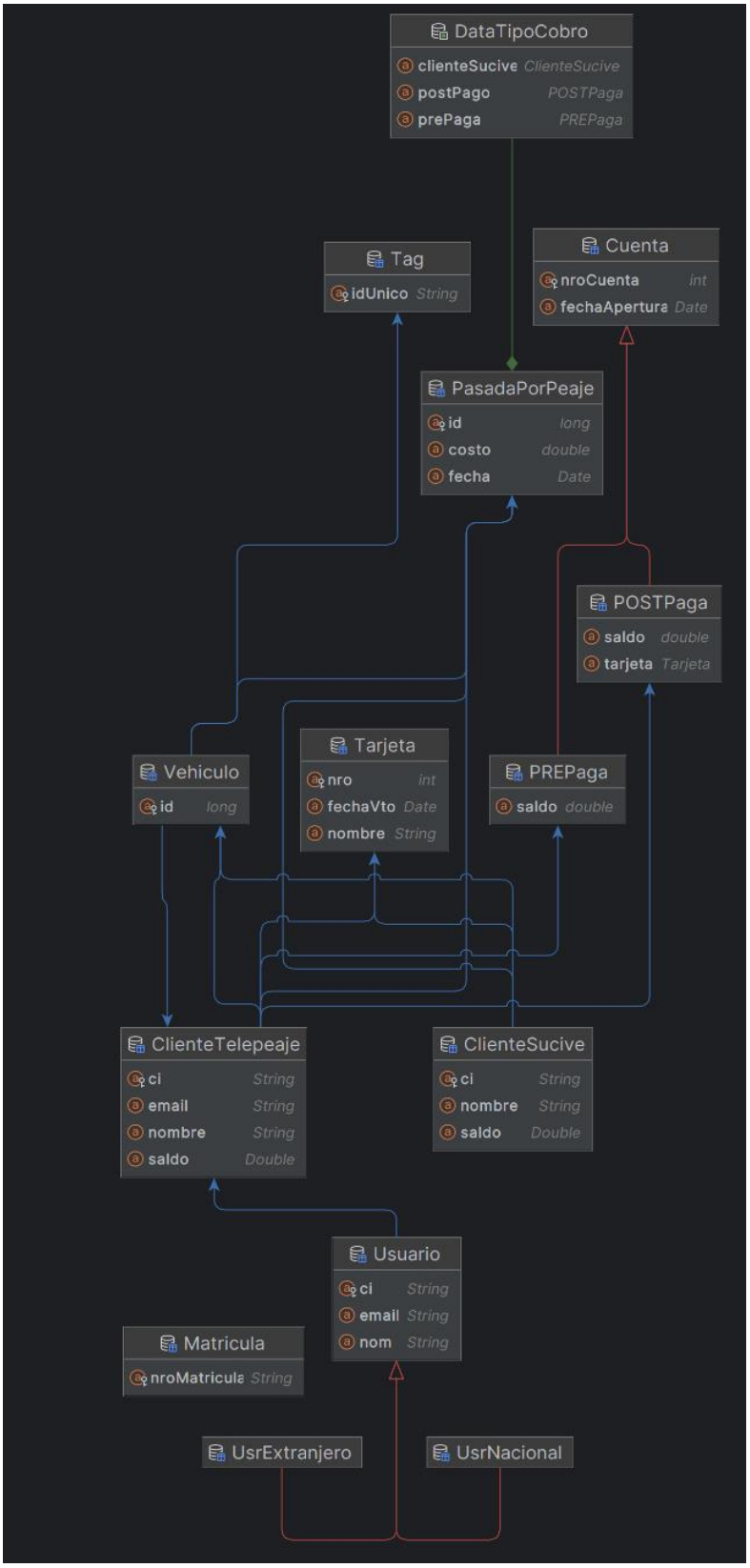
6 Persistencia

6.1 Modelado JPA

6.1.1 Diagrama de Modulo Peaje

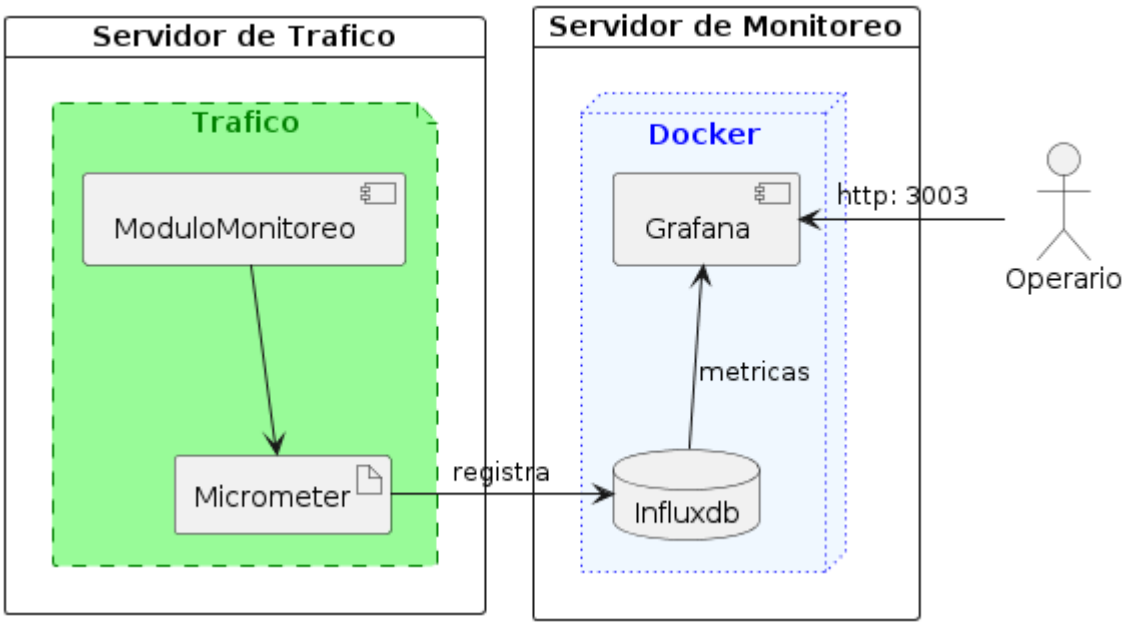


6.1.2 Diagrama de Modulo Gestión Cliente



7 Doker, Grafana e InfluxDB

7.1 Arquitectura implementada

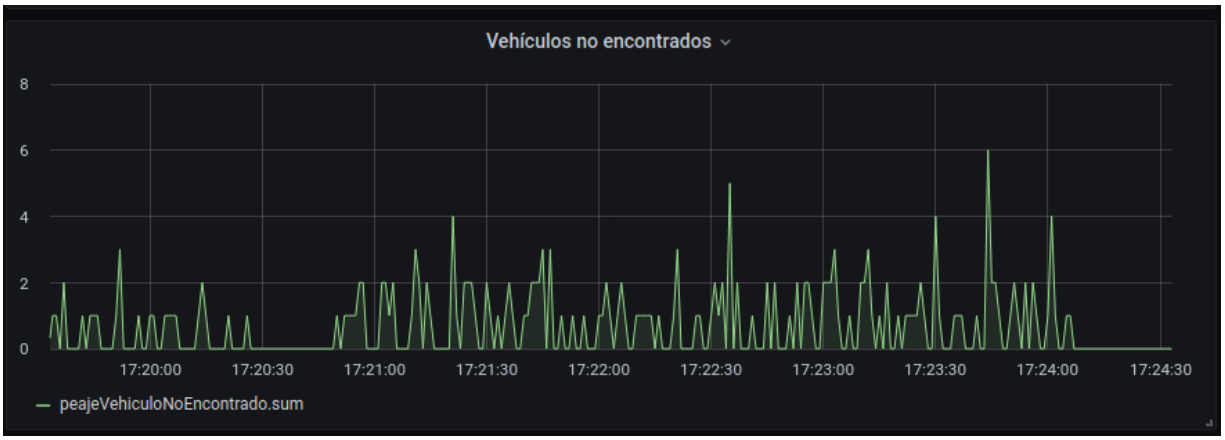


Continuando con el desarrollo de nuestro trabajo utilizando doker generamos un contenedor para manejar grafana e influxdb para monitorear los datos de nuestra aplicación.

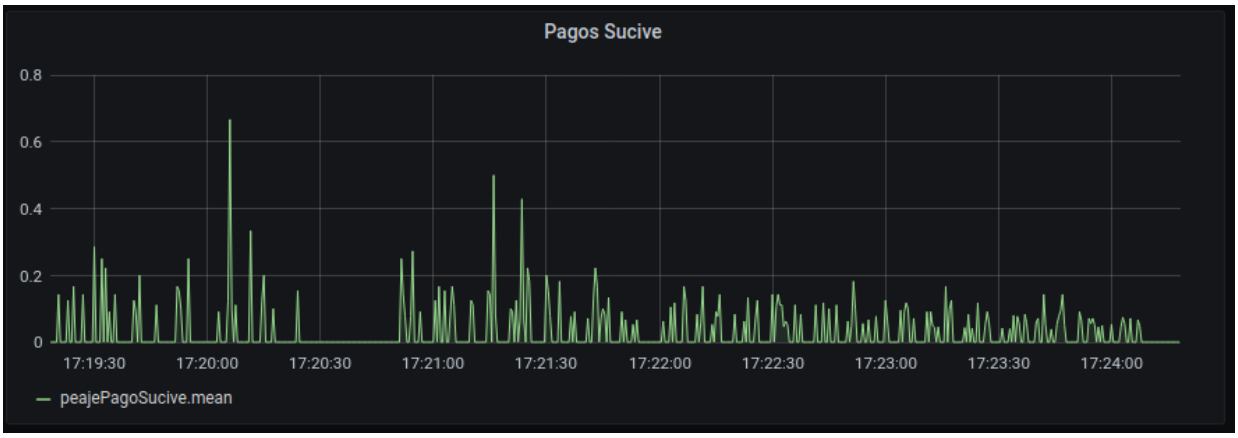
7.2 Gráficos de Grafana

A continuación, se mostrarán las gráficas con las métricas de grafana, así como también la persistencia de influxdb. Usamos un script para generar tráfico y poder simular de manera más orgánica las métricas.

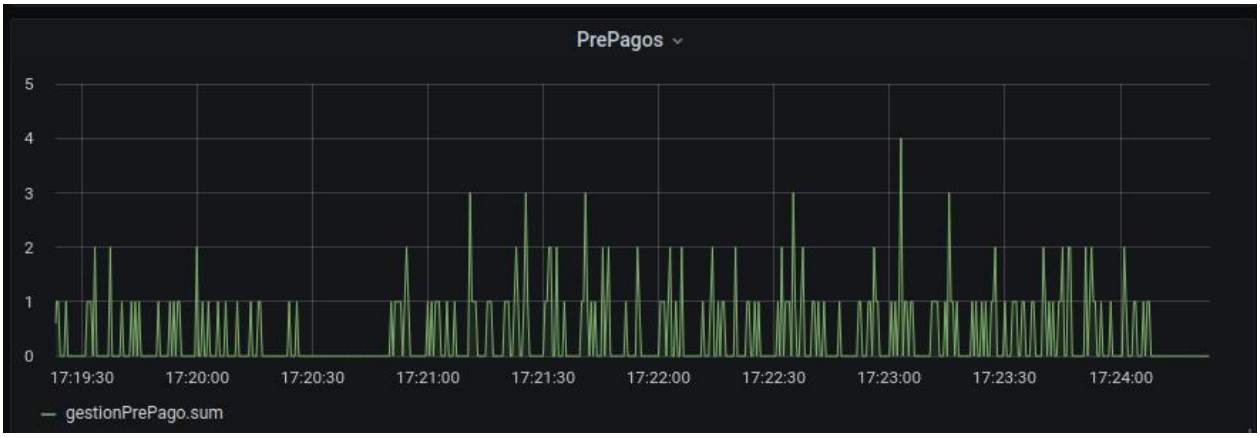
7.2.1 Vehículo No Encontrado



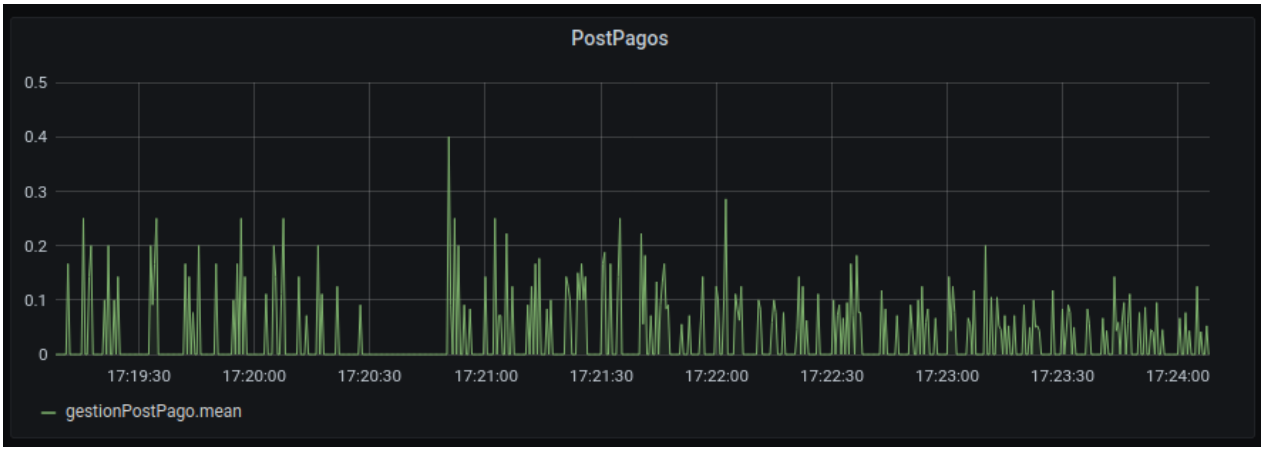
7.2.2 Pagos Sucive



7.2.3 Pre Pagos



7.2.4 Post Pagos



7.3 Influxdb

7.3.1 Tablas de la base de datos

```
carlos@carlos-VirtualBox:~/Escritorio$ influx -host 127.0.0.1 -port 8086
Connected to http://127.0.0.1:8086 version 1.8.2
InfluxDB shell version: 1.6.7~rc0
> use tallerjavadb
Using database tallerjavadb
> show measurements;
name: measurements
name
----
gestionPostPago
gestionPrePago
peajePagoSucive
peajeVehiculoNoEncontrado
>
```

7.3.2 Contenido de las tablas

Consulta Post Pago:

```
> select * from gestionPostPago limit 10;
name: gestionPostPago
time                metric_type value
----                ----------- -----
1718129543824000000 counter     1
1718129551638000000 counter     1
1718129551841000000 counter     1
1718129553404000000 counter     1
1718129553604000000 counter     1
1718129553771000000 counter     0
1718129555321000000 counter     1
1718129555595000000 counter     1
1718129560320000000 counter     1
1718129560705000000 counter     1
>
```

Consulta Pre Pago:

```
name: gestionPrePago
time                metric_type value
----                ----------- -----
1718222424730000000 counter     1
1718222434655000000 counter     0
1718222444657000000 counter     0
1718222454656000000 counter     0
1718222464656000000 counter     0
1718222474666000000 counter     0
1718222484662000000 counter     0
1718222494658000000 counter     0
1718222504670000000 counter     0
1718222514657000000 counter     0
```

Consulta Pago Sucive:

```
> select * from peajePagoSucive limit 10;
name: peajePagoSucive
time                metric_type value
----                ----------- -----
1718146207381000000 counter     1
1718146214178000000 counter     1
1718146214278000000 counter     1
1718146217355000000 counter     0
1718146224171000000 counter     0
1718146224273000000 counter     0
1718146227355000000 counter     0
1718146234174000000 counter     0
1718146234275000000 counter     0
1718146237355000000 counter     0
```

Consulta Vehículo No Encontrado:

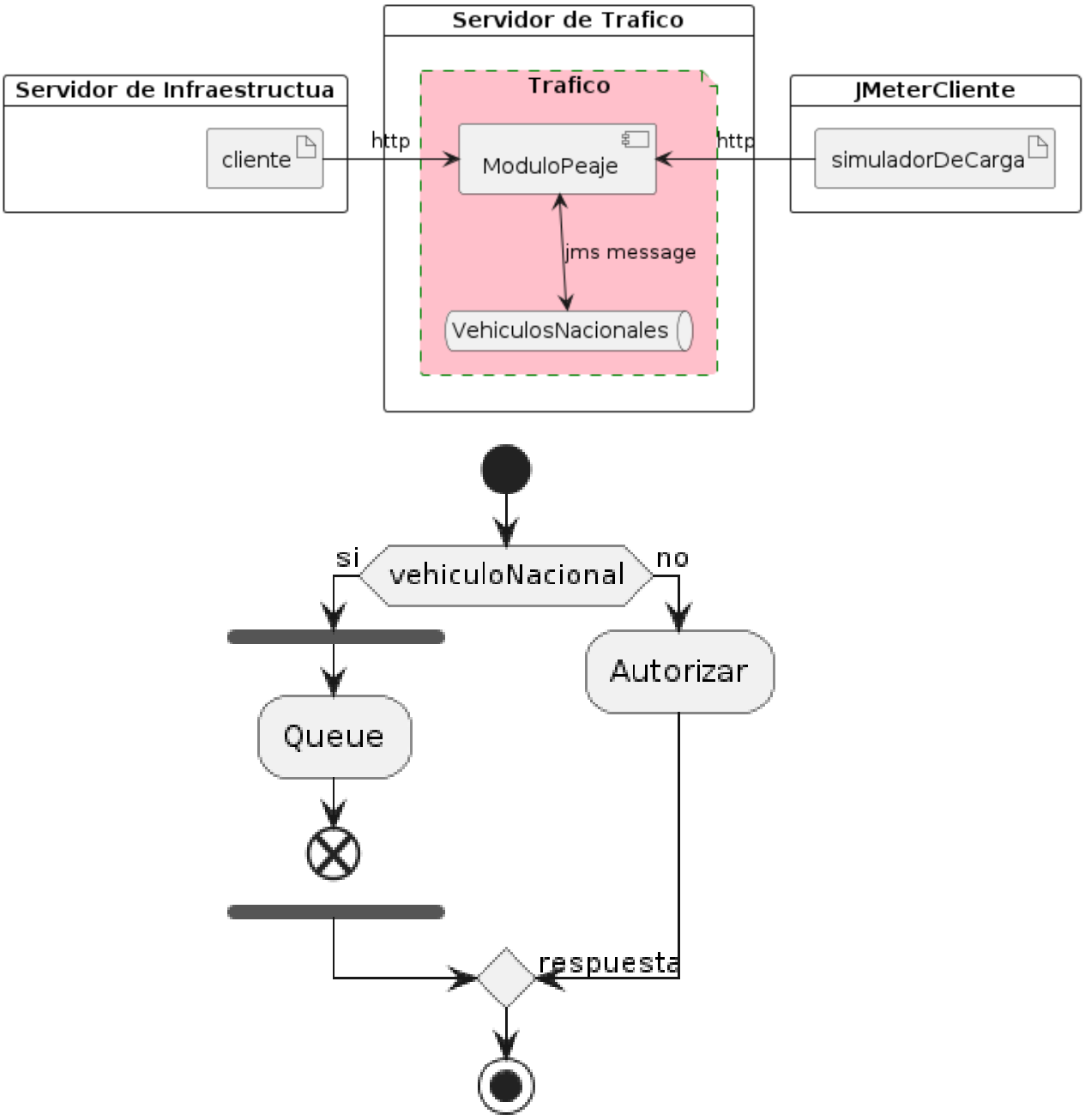
```
name: peajeVehiculoNoEncontrado
time                metric_type value
----                ----------- -----
1718138782056000000 counter     1
1718138792023000000 counter     0
1718138795601000000 counter     1
1718138802021000000 counter     0
1718138805602000000 counter     0
1718138812021000000 counter     0
1718138815599000000 counter     0
1718138822028000000 counter     0
1718138825608000000 counter     0
1718138831192000000 counter     1
```

7.4 *Script generador de trafico*

```
#!/bin/bash

duracion=60

# Función para realizar los curls
curls(){
    curl -X GET -v http://localhost:8080/Sistema-Gestion/rest/autorizar -H "Content-Type: application/json" -d '{"tag":11111,"matricula":"BAA 1234"}'
    curl -X GET -v http://localhost:8080/Sistema-Gestion/rest/Pago-Sucive -H "Content-Type: application/json" -d '{"matricula":"BAA 1234","importe":1}'
    curl -X GET -v http://localhost:8080/Sistema-Gestion/rest/Gestion/PrePago -H "Content-Type: application/json" -d '{"tag":"C3","importe":1}'
    curl -X GET -v http://localhost:8080/Sistema-Gestion/rest/Gestion/PostPago -H "Content-Type: application/json" -d '{"tag":"C3","importe":1}'
}


for ((i=0; i<$duracion; i++)); do
    curls
    tiempo_aleatorio=$(shuf -i 0-2 -n 1)
    sleep $tiempo_aleatorio
done

```

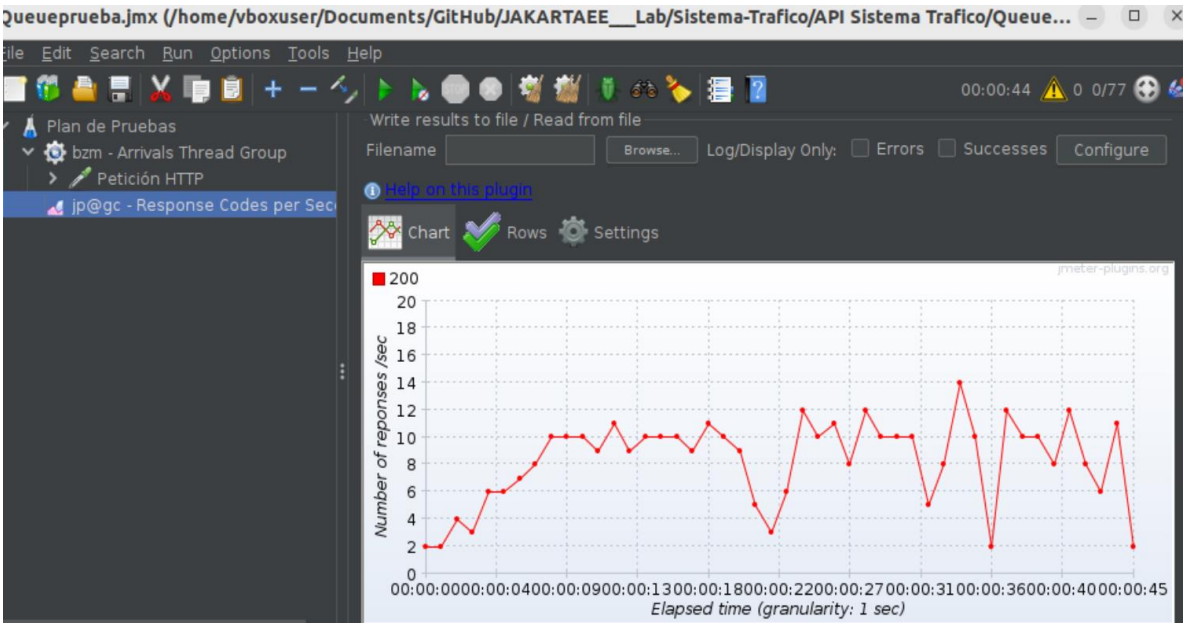
8 Jakarta Messaging

8.1 Diagrama de arquitectura



8.2 Queue

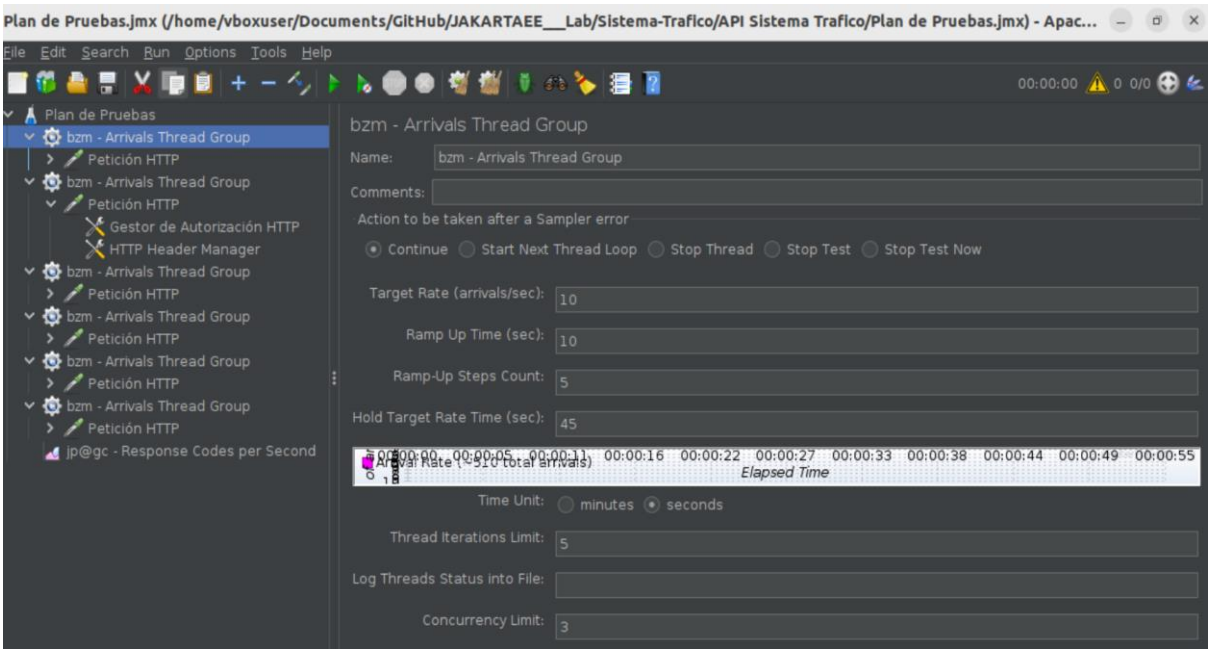
Se implementó para procesar vehículo nacional una Queue de mensajes JMS de tipo point to point para gestionar la cantidad de peticiones en cierto periodo de tiempo.



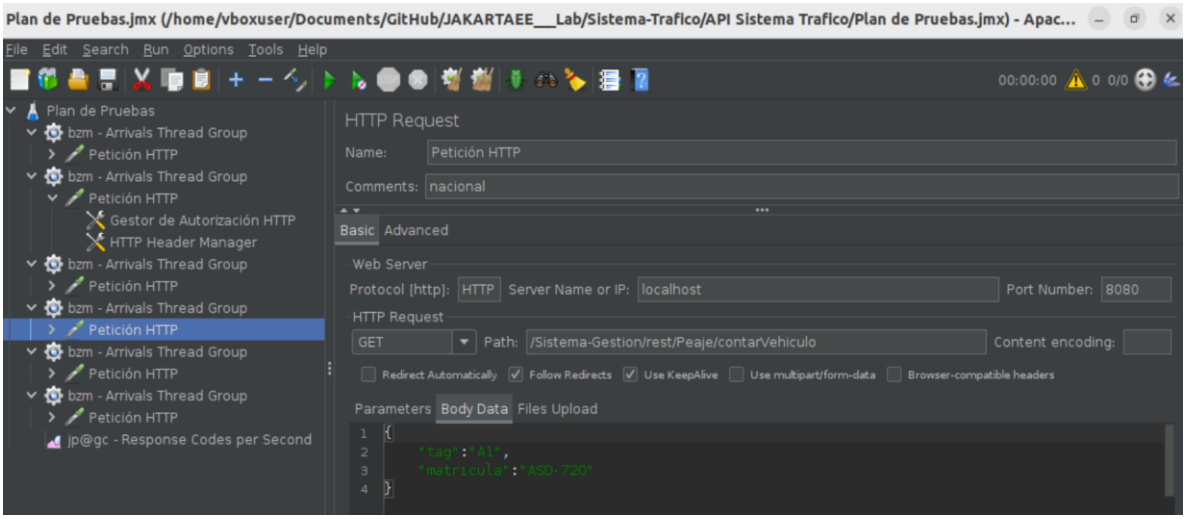
En la gráfica podemos observar como cuando se excede la cantidad de peticiones el flujo de datos se detiene y luego de un tiempo vuelve a procesar.

8.3 Test de carga con JMeter

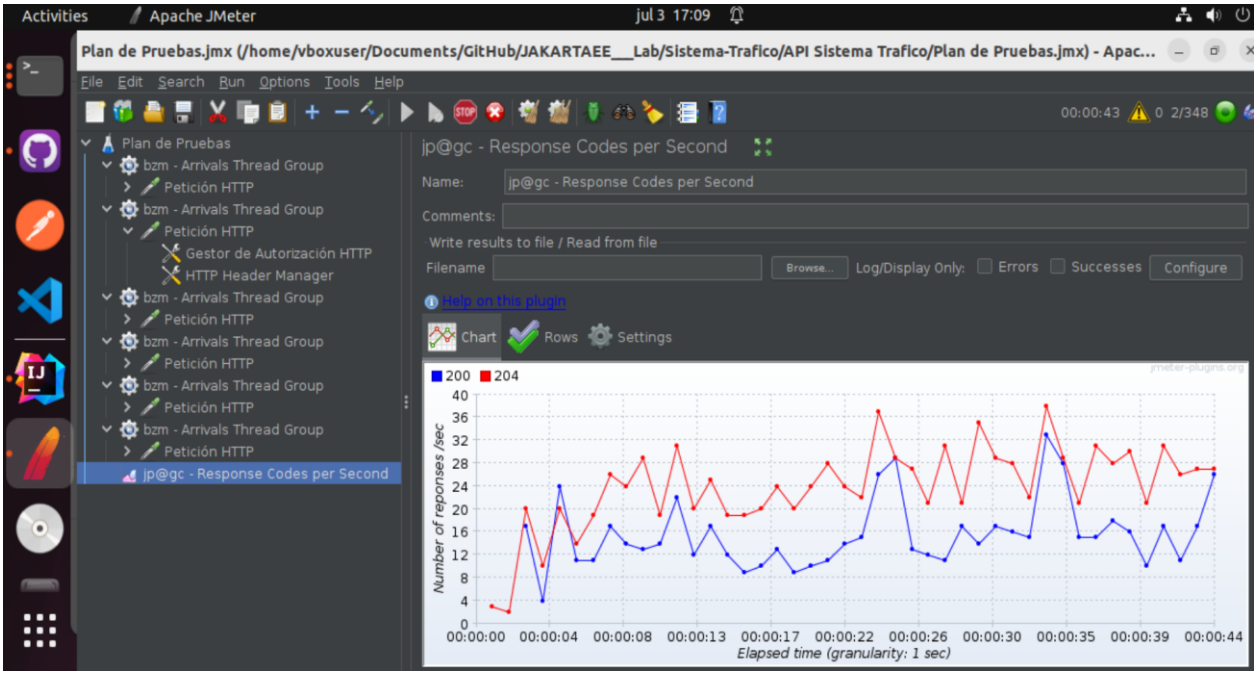
Usando jmeter realizamos diferente carga utilizando los diferentes endpoints con los que veníamos trabajando.



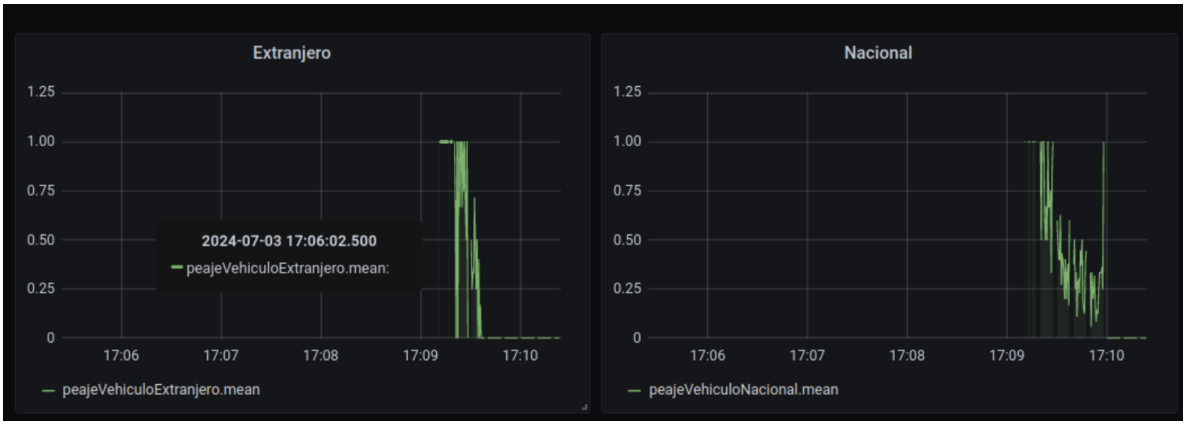
Por ejemplo, en el siguiente caso se configuro la aplicación con el endpont para contar vehículos y se le pasa en el body los datos de un vehículo nacional.

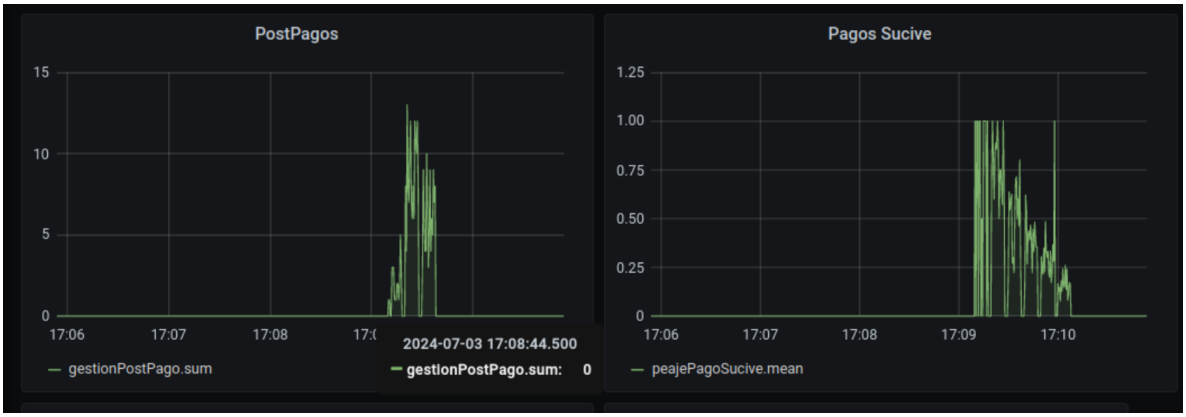


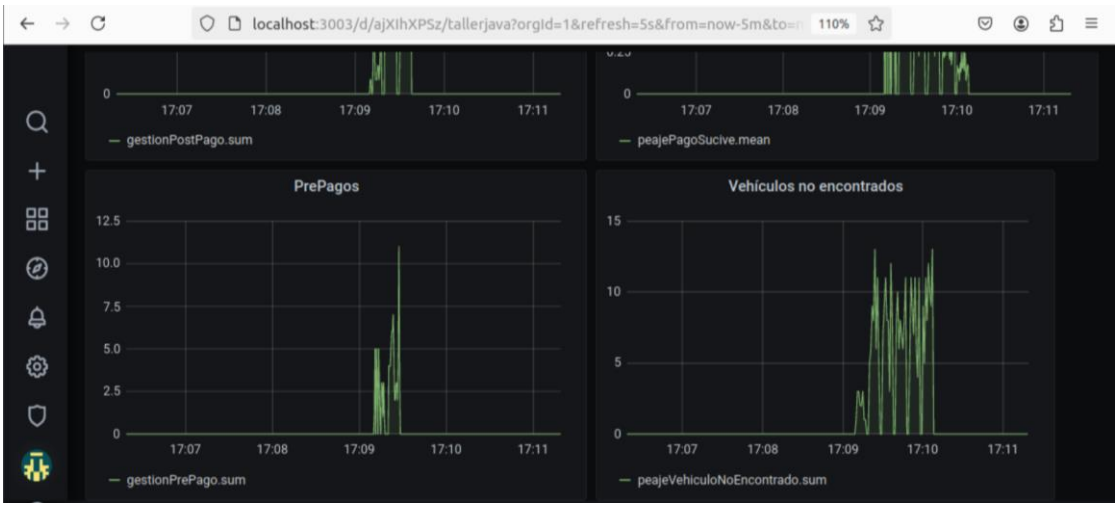
Una vez configurado todo para las pruebas iniciamos con el plan y la gráfica refleja los siguientes datos:



Al correr jmeter observamos el impacto en grafana:







9 Infraestructura Utilizada

- Versión de Java 17 Temurin
- Intellij(IDE)
- JUnit
- Wildfly
- Eclipselink
- MariaDB
- JPA
- JMS
- Loombok
- Grafana
- Influxdb
- Docker