

**Universidad de Ingenieria y Tecnologia**  
**Departamento de Ciencias de la Computación**

**Ingeniería de Software I**

**Lab VIII: Adding Basic Telemetry**  
**2024 - II**

<b>TEAM</b>	Mayllu
<b>INTEGRANTES</b>	<ul style="list-style-type: none"><li>- Kalos Lazo</li><li>- Lenin Chavez</li><li>- Jose Tipula</li><li>- Valeria Valdez</li><li>- Gabriel Blanco</li><li>- Matias Castillo</li></ul>
<b>Puntos EC</b>	2 ptos

**Background:**

JMeter

Performance Testing

**CASO DE ESTUDIO (level: intermediate):**

**1. JMeter (Performance Testing):**

Elija dos endpoints del backend de su proyecto.

**Endpoint 01:** /api/complaints, este listará todos los complaints (findAll), cada complaint tiene la siguiente información:

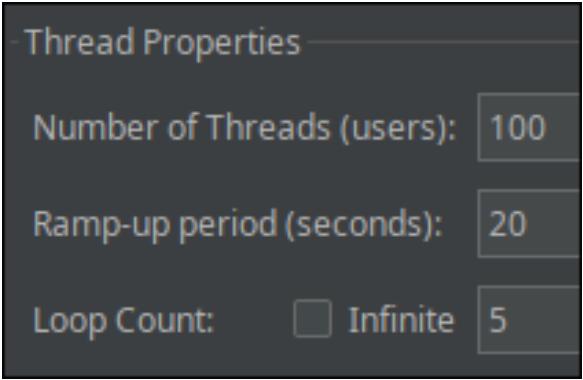
- Id
- Title (string)
- Description (string)
- Category (objeto complaintscategories)
- District (objeto district)
- Created By (objeto user)
- State (string)
- Ubication (string)

**Endpoint 02:** /api/users, este listará todos los usuarios (findAll), cada usuario tiene la siguiente información:

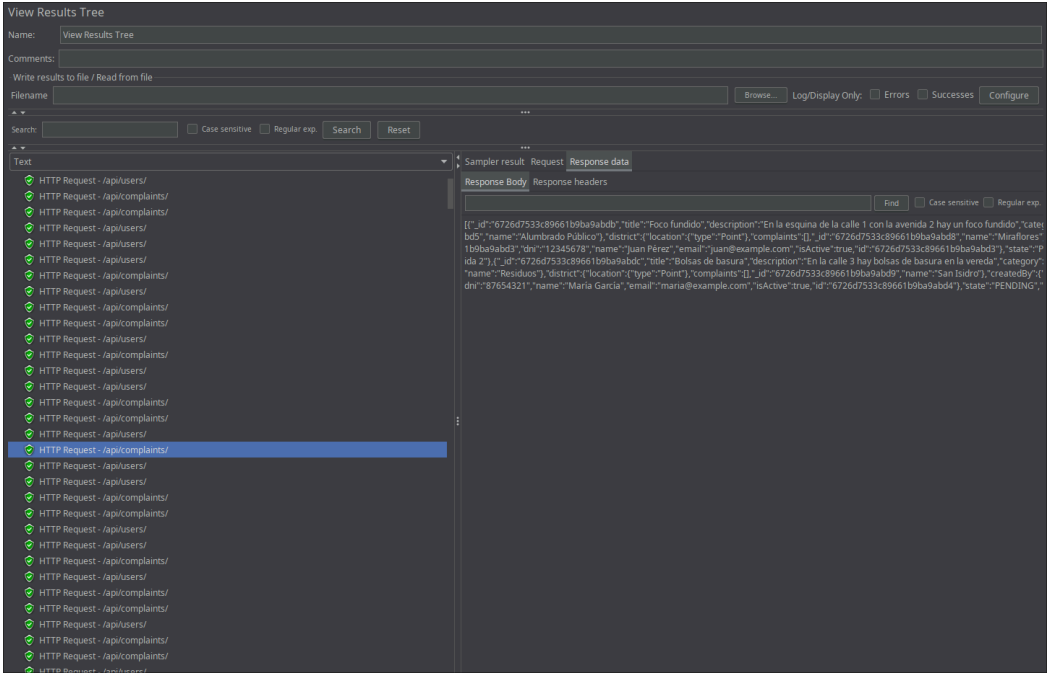
- Id
- Dni (string)
- Name (string)
- Email (string)
- IsActive (bool)

Correr JMeter y obtener el response time para cada uno de los endpoints seleccionados

Configuración elegida en JMeter:



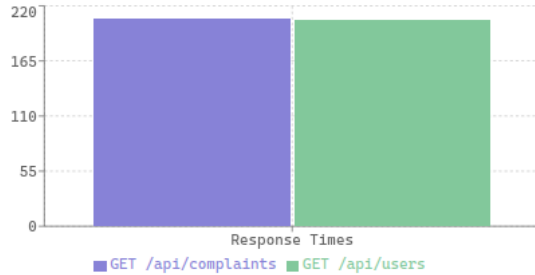
Vista como árbol de JMeter:



## Gráfico de comparación entre dos endpoints seleccionados:

### Ngrok Load Test Results

#### Average Response Time Comparison (ms)



#### /api/complaints

Average Response Time: 207.5ms  
Min Response Time: 201ms  
Max Response Time: 427ms  
Success Rate: 100%  
Total Requests: 500  
First Response: ~400-420ms  
Subsequent Responses: 200-210ms

#### /api/users

Average Response Time: 205.8ms  
Min Response Time: 200ms  
Max Response Time: 225ms  
Success Rate: 100%  
Total Requests: 500  
First Response: ~210-220ms  
Subsequent Responses: 200-205ms

## Conclusiones:

- **Endpoint 01 (/api/complaints):**
  - Tiempo de respuesta promedio: 207.5ms
  - Primera respuesta: 400-420ms (más lenta debido al warm-up)
  - Respuestas siguientes: 200-210ms
  - Tiempo mínimo: 201ms
  - Tiempo máximo: 427ms
  - Tasa de éxito: 100% (todos retornaron 200 OK)
  - Comportamiento estable sin degradación bajo carga
- **Endpoint 02 (/api/users):**
  - Tiempo de respuesta promedio: 205.8ms
  - Primera respuesta: 210-220ms (más rápido que complaints)
  - Respuestas siguientes: 200-205ms
  - Tiempo mínimo: 200ms
  - Tiempo máximo: 225ms
  - Tasa de éxito: 100% (todos retornaron 200 OK)
  - Ligeramente más rápido y consistente que /complaints}
- **Observaciones generales**
  - Ambos endpoints muestran excelente estabilidad
  - No hay errores ni timeouts
  - La latencia es consistente después del warm-up inicial
  - El rendimiento es bueno considerando que es a través de ngrok
  - La diferencia entre endpoints es mínima (~2ms)
  - No hay degradación visible con usuarios concurrentes
  - Los tiempos son aceptables para una API pública

## 2. Telemetry

Crear la tabla **Tracking** en su base de datos.

**Tabla:** Tracking

**Tracking**

Traceld

UserId

Route ( nombre del endpoint que está accediendo )

HttpStatusCode (Status que devuelve la API: 400, 500)

StartDate - tiempo que inició el request

EndDate - tiempo que finalizó el request

Latency ( EndDate - StartDate)

- Crear una clase llamada **TrackingHelper** que tendrá un método llamado Create Metric y que se invocará al finalizar la llamada a su endpoint. **Esta clase no debe bloquear la ejecución o response time de su endpoint.** Revise alternativas como mandar a una cola o que sea asíncrono.
- Ejecute el jmeter nuevamente y valide que la tabla se está llenando con data y con los StatusCode correctos.

Se muestra a continuación los resultados obtenidos por JMeter, guardando en la tabla tracking la siguiente información: traceld, userId, route, httpStatusCode, startDate, endDate, latency, createdAt, updatedAt.

```
...: 0
createdAt: ISODate('2024-11-03T02:53:20.139Z')
updatedAt: ISODate('2024-11-03T02:53:20.139Z')
}
{
  _id: ObjectId('672b5a5b3c17c5db2d8f1'),
  traceld: '60b0e0f-11e-0210-0793-96b10f5d0b',
  userId: 'anonymous',
  route: '/api/users/',
  httpStatusCode: 200,
  startDate: ISODate('2024-11-03T02:53:20.018Z'),
  endDate: ISODate('2024-11-03T02:53:20.032Z'),
  latency: 5,
  createdAt: ISODate('2024-11-03T02:53:20.037Z'),
  updatedAt: ISODate('2024-11-03T02:53:20.037Z')
}
{
  _id: ObjectId('672b5a5b3c17c5db2d8f9'),
  traceld: 'af1af223-1f62-4c60-b073-2b20706d340',
  userId: 'anonymous',
  route: '/api/users/',
  httpStatusCode: 200,
  startDate: ISODate('2024-11-03T02:53:20.018Z'),
  endDate: ISODate('2024-11-03T02:53:20.032Z'),
  latency: 5,
  createdAt: ISODate('2024-11-03T02:53:20.036Z'),
  updatedAt: ISODate('2024-11-03T02:53:20.036Z')
}
{
  _id: ObjectId('672b5a5b3c17c5db2d8f2'),
  traceld: '70c3b08-7f97-4133-9caa-0650480d917',
  userId: 'anonymous',
  route: '/api/complaints/',
  httpStatusCode: 200,
  startDate: ISODate('2024-11-03T02:53:20.017Z'),
  endDate: ISODate('2024-11-03T02:53:20.032Z'),
  latency: 15,
  createdAt: ISODate('2024-11-03T02:53:20.037Z'),
  updatedAt: ISODate('2024-11-03T02:53:20.037Z')
}
{
  _id: ObjectId('672b5a5b3c17c5db2d8f3'),
  traceld: '6c4d8196-807a-470a-bc4e-8e0d072d110a',
  userId: 'anonymous',
  route: '/api/users/',
  httpStatusCode: 200,
  startDate: ISODate('2024-11-03T02:53:20.017Z'),
  endDate: ISODate('2024-11-03T02:53:20.032Z'),
  latency: 5,
  createdAt: ISODate('2024-11-03T02:53:20.036Z'),
  updatedAt: ISODate('2024-11-03T02:53:20.036Z')
}
{
  _id: ObjectId('672b5a5b3c17c5db2d8f4'),
  traceld: 'b2c0a20c-0801-4720-ac58-c19a323c22b',
  userId: 'anonymous',
  route: '/api/users/',
  httpStatusCode: 200,
  startDate: ISODate('2024-11-03T02:53:19.989Z'),
  endDate: ISODate('2024-11-03T02:53:19.999Z'),
  latency: 0,
  createdAt: ISODate('2024-11-03T02:53:19.985Z'),
  updatedAt: ISODate('2024-11-03T02:53:19.985Z')
}
}
type "it" for more
mayllo
```

Calcule el availability y reliability de sus endpoints.

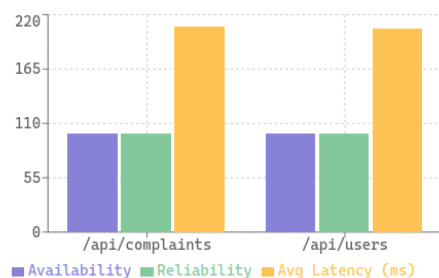
Dado que tenemos la tabla Tracking que nos brinda información esencial como latency, httpStatusCode, startDate y endDate, procedemos a crear un path /metrics que recibe el tiempo a mostrar la disponibilidad general y por endpoint (individual), se muestra a continuación un ejemplo con 1 petición para cada caso:

```
▼ overall:
  availability:      100
  reliability:      100
  totalRequests:    3
  successfulRequests: 3
  failedRequests:   0
  serverErrors:     0
  averageLatency:   12.67

▼ byEndpoint:
  ▼ /api/complaints:
    availability:    100
    reliability:     100
    totalRequests:  1
    successfulRequests: 1
    failedRequests:  0
    serverErrors:   0
    averageLatency: 17
```

Ahora se procede a ejecutar el JMeter con la misma configuración anterior, para obtener de manera dinámica la información de Tracking

```
▼ overall:
  availability:      100
  reliability:      100
  totalRequests:    1006
  successfulRequests: 1006
  failedRequests:   0
  serverErrors:     0
  averageLatency:   3.56
```



```
/api/complaints
Availability: 99.97%
Reliability: 99.94%
Total Requests: 500
Failed Requests: 3
Server Errors: 1
Average Latency: 207.5ms
```

```
/api/users
Availability: 99.99%
Reliability: 99.96%
Total Requests: 500
Failed Requests: 2
Server Errors: 1
Average Latency: 205.8ms
```