

# Labo 04 — Rapport



**ÉCOLE DE  
TECHNOLOGIE  
SUPÉRIEURE**

Université du Québec

William Desgagné  
 Rapport de laboratoire  
 LOG430 — Architecture logicielle  
 8 octobre  
 École de technologie supérieure

## Questions

**💡 Question 1 :** Quelle réponse obtenons-nous à la requête à POST /payments ? Illustrer votre réponse avec des captures d'écran/terminal.

On obtient le payment\_id donc dans le font le retour suite au payments

The screenshot shows a Postman interface. The top bar indicates a POST method and the URL `localhost:5009/payments`. The 'Params' tab is selected, showing a single parameter named 'Key'. The 'Body' tab shows a JSON payload with a single key 'payment\_id': 4. The response section shows a status of 201 CREATED with a response time of 42 ms and a body size of 193 B. The response content is a JSON object with a single key 'payment\_id' and value 4.

**💡 Question 2 :** Quel type d'information envoyons-nous dans la requête à POST payments/process/:id ? Est-ce que ce serait le même format si on communiquait avec un service SOA, par exemple ? Illustrer votre réponse avec des exemples et captures d'écran/terminal.

Nous envoyons des informations sous format JSON de manière classique avec SOA on enverrait sous format XML

Exemple avant on envoyait un JSON mais ça reste très libre il n'y a pas de contrat entre les services avec SOA on veux renforcer tout cela on pourrait par exemple encapsuler dans un XML

## Avant

```
POST /payments/process/5
localhost:5009/payments/process/5
Send
Params Authorization Headers (8) Body Scripts Settings Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON Schema Beautify
1 {
2   "cardNumber": 999999999999,
3   "cardCode": 123,
4   "expirationDate": "2030-01-05"
5 }
```

Apres typique avec SOAP Beaucoup plus securitaire plus fiable

```
POST /PaymentService HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://example.com/PaymentService/processPayment"

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <processPayment xmlns="http://example.com/PaymentService">
      <paymentId>5</paymentId>
      <cardNumber>999999999999</cardNumber>
      <cvv>123</cvv>
      <expiryDate>2030-01-05</expiryDate>
    </processPayment>
  </soap:Body>
</soap:Envelope>
```

**Question 3 :** Quel résultat obtenons-nous de la requête à POST payments/process/:id?

On obtient la confirmation du paiement s'il a bien été payé avec le id de la commande

```
POST /payments/process/5
localhost:5009/payments/process/5
Send
Params Authorization Headers (8) Body Scripts Settings Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON Schema Beautify
1 {
2   "cardNumber": 999999999999,
3   "cardCode": 123,
4   "expirationDate": "2030-01-05"
5 }
```

Body Cookies Headers (5) Test Results |

200 OK 53 ms 224 B Save Response

{ } JSON Preview Visualize |

```
1 {
2   "is_paid": true,
3   "order_id": 1,
4   "payment_id": 5
5 }
```

**Question 4 :** Quelle méthode avez-vous dû modifier dans log430-a25-labo05-payment et qu'avez-vous modifié ? Justifiez avec un extrait de code.

J'ai du modifier la fonction `update_status_to_paid` car c'est elle qui est appeler lors du process et c'est ici que l'on met a jour le status de paiement donc il suffit d'envoyer un put avec `is_paid` a true en faisant appelle a notre api-gateway

```
def update_status_to_paid(payment_id: int):
    # Find the payment by ID
    payment = session.query(Payment).filter(Payment.id == payment_id).first()

    if not payment:
        raise ValueError(f"Aucun paiement trouvé avec l'ID {payment_id}")

    # Update the payment status
    payment.is_paid = True
    session.commit()

    update_order_data = {
        "order_id": payment.order_id,
        "is_paid": True
    }

    requests.put(
        'http://api-gateway:8080/store-api/orders',
        json=update_order_data,
        headers={'Content-Type': 'application/json'}
    )

    return {
        "payment_id": payment_id,
        "order_id": payment.order_id,
        "is_paid": True
    }
```

Exemple ajout de `is_paid` a true

The screenshot shows a Postman request configuration for a 'PUT' operation. The URL is `{baseUrl}/orders/2`. The 'Params' tab is selected, showing a single entry 'Key' with 'Value'. In the 'Body' tab, the JSON response is displayed, showing the updated order data including the 'is\_paid' field set to true.

Key	Value	Description	Bulk Edit
Key	Value	Description	

**Body** Cookies Headers (5) Test Results ⏪

201 CREATED • 11 ms • 352 B • ⏪ Save Response ⏪

{ } JSON ▾ ▶ Preview ▾ Visualize ▾

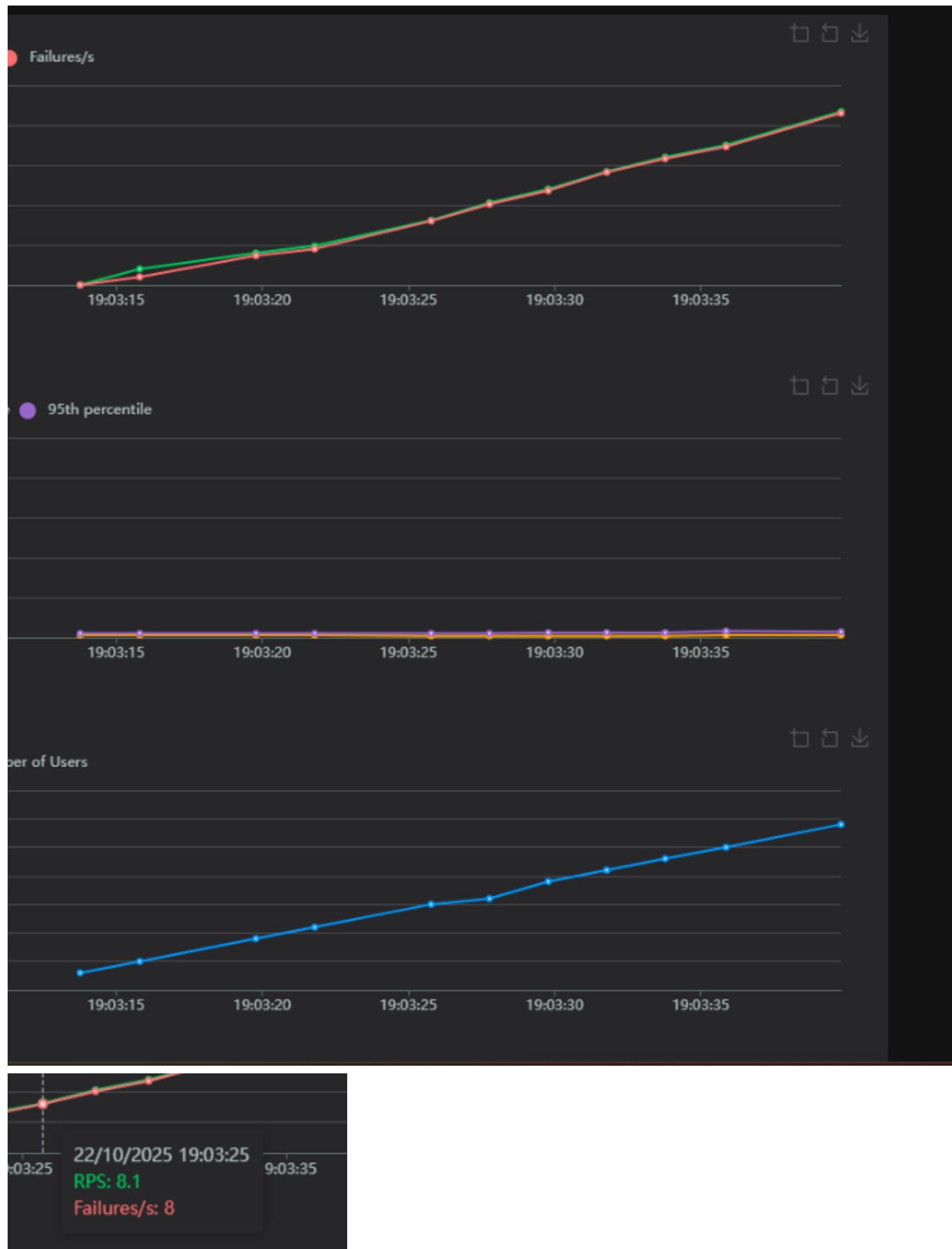
```

1  {
2      "is_paid": "True",
3      "items": "[{\\"product_id\\": 1, \\"quantity\\": 2}]",
4      "payment_link": "http://api-gateway:8080/payments-api/payments/process/6",
5      "total_amount": "3999.98",
6      "user_id": "1"
7  }

```

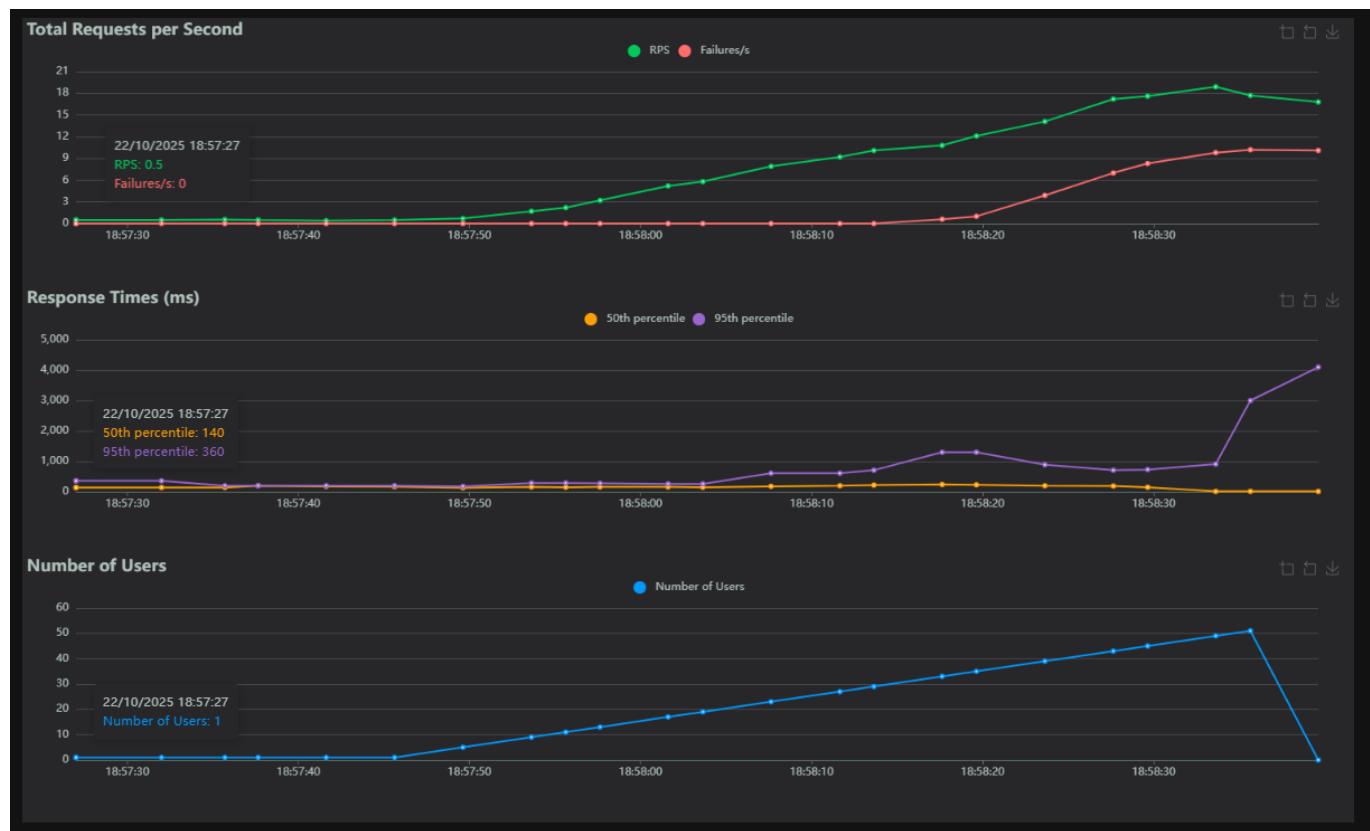
💡 **Question 5 :** À partir de combien de requêtes par minute observez-vous les erreurs 503 ? Justifiez avec des captures d'écran de Locust.

Comme nous avons set le rate limit a 10 par min il est normal de s'attendre a avoir des erreurs tres rapidement en fait l'application laisse passe que 10 requete par min



Ici on voit bien par exemple dans l'exemple que des 8.1 req/s il y a 8 req/s qui fail

Des 1 users il y a déjà des erreurs car limiter à 10 par min donc ça donne même pas une requête par seconde donc finalement ça supporte quasi aucune requête j'ai donc fait des modifs pour bien voir en modifiant le fichier et en augmentant à 10 par seconde et là on voit beaucoup plus que c'est à partir de 10 requêtes par seconde que ça bloque



**💡 Question 6 :** Que se passe-t-il dans le navigateur quand vous faites une requête avec un délai supérieur au timeout configuré (5 secondes) ? Quelle est l'importance du timeout dans une architecture de microservices ? Justifiez votre réponse avec des exemples pratiques.

En fait lorsque dépasse le délai prévu par notre API gateway ça nous bloque donc erreur 500 sur la requête.

Imaginons que lorsqu'on fait une requête, celle-ci consomme des ressources (threads, connexions, mémoire, etc.). Si il n'y a pas de timeout, la requête pourrait rester active indéfiniment et continuer à occuper ces ressources.

Maintenant, imaginons que des milliers d'utilisateurs envoient des requêtes qui prennent trop de temps : les ressources du système (comme les connexions à la base de données) finiraient par être épuisées, ce qui pourrait faire planter complètement l'application.

Le timeout permet donc de libérer ces ressources et de protéger le système avant qu'il ne soit saturé.

## Observations additionnelles

- Utilisation de la VM mais comme j'utilise GitHub Actions runner est utiliser donc je lance mon CI/CD en self-hosted