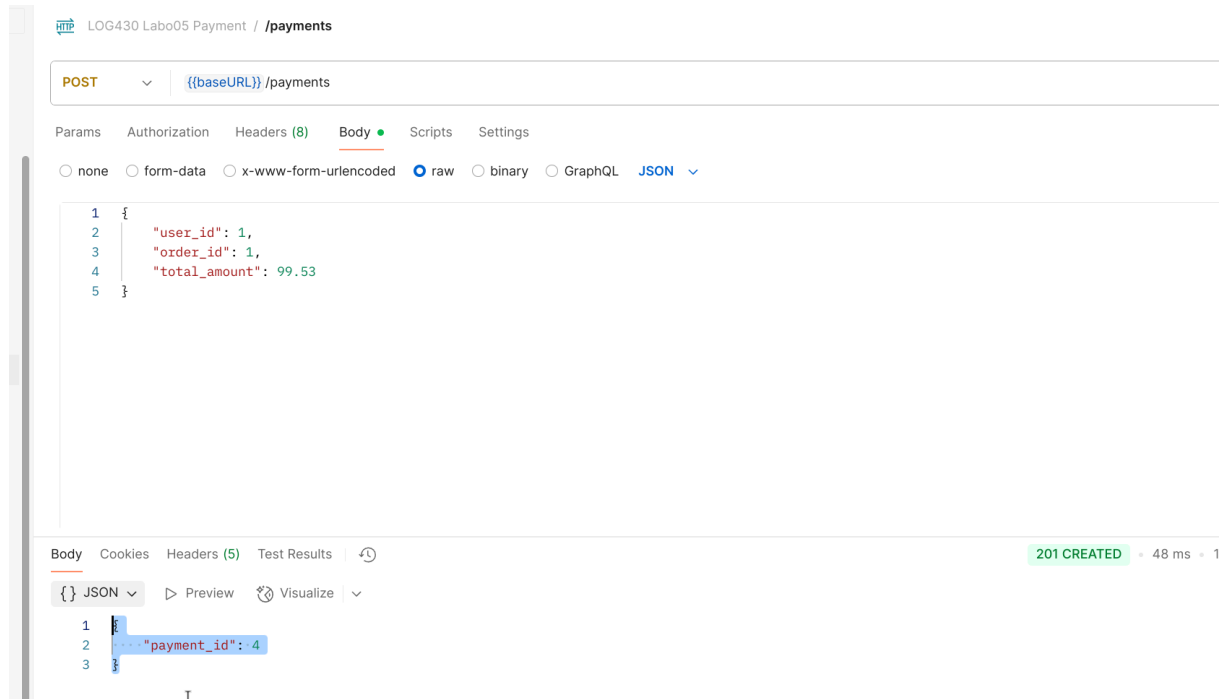


log430-a25-labo05 - Rapport Louis Thevenet

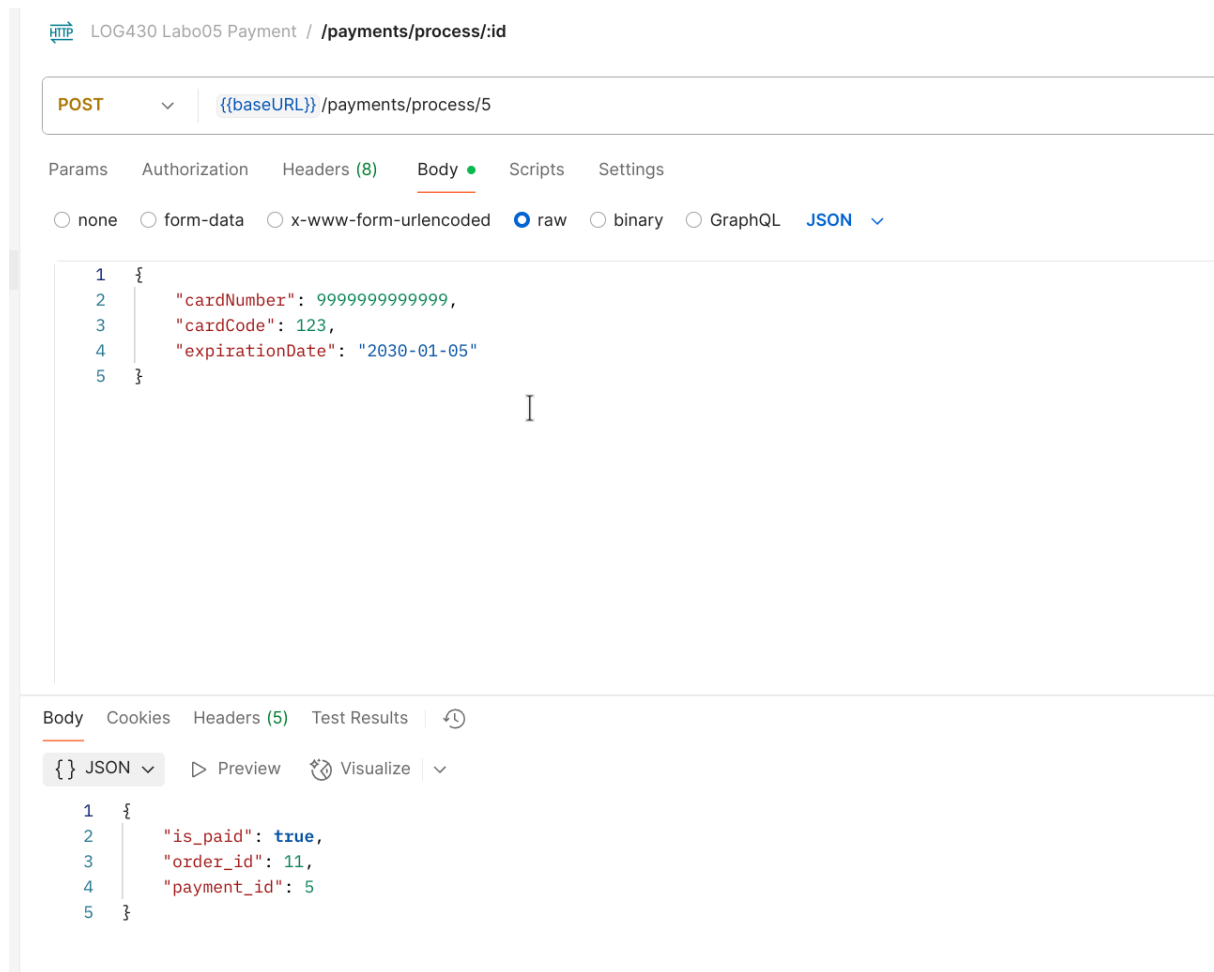
Question 1

Quelle réponse obtenons-nous à la requête à POST /payments ? Illustrez votre réponse avec des captures d'écran/terminal. Le service de paiement renvoie un identifiant de paiement.



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 ? Illustrez votre réponse avec des exemples et captures d'écran/terminal.



On transmet les information bancaire dans le corps de la requête au format JSON.

Si on communiquait avec un service SOA, on utiliserait un format plus standardisé comme SOAP, qui utilise XML pour structurer les données.

Question 3

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

Voir capture d'écran précédente. on obtient des informations de confirmation de paiement, y compris un identifiant de transaction, le statut du paiement et le montant payé.

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 modifié la méthode `process_payment` pour notifier le système `store_manager` que la commande est payée après avoir traité le paiement. J'ai ajouté une requête HTTP PUT pour mettre à jour le statut de la commande dans le système `store_manager`.

```
def process_payment(payment_id, credit_card_data):
    """ Process payment with given ID, notify store_manager sytem that the order is
    paid """
    # S'il s'agissait d'une véritable API de paiement, nous enverrions les données de
    la carte de crédit à un payment gateway (ex. Stripe, Paypal) pour effectuer le
    paiement. Cela pourrait se trouver dans un microservice distinct.
```

```

_process_credit_card_payment(credit_card_data)

# Si le paiement est réussi, mettre à jour les statut de la commande
# Ensuite, faire la mise à jour de la commande dans le Store Manager (en
utilisant l'order_id)
update_result = update_status_to_paid(payment_id)

#####
# Notify the store manager system
store_manager_api_url = "http://krakend:8080/store-api/orders"
payload = {"order_id": update_result['order_id'], "is_paid": True}
response = requests.put(store_manager_api_url, json=payload)
#####

print(f"Updated order {update_result['order_id']} to paid={update_result}")
result = {
    "order_id": update_result["order_id"],
    "payment_id": update_result["payment_id"],
    "is_paid": update_result["is_paid"]
}

return result

```

```

store_manager | 172.19.0.9 - - [22/Oct/2025 14:38:40] "GET /metrics HTTP/1.1" 200 -
store_manager | 2025-10-22 14:38:42 - order_controller - DEBUG - Mettre à jour la commande 1, status=True
store_manager | 2025-10-22 14:38:42 - order_controller - DEBUG - Statut actuel True
store_manager | 172.19.0.6 - - [22/Oct/2025 14:38:42] "PUT /orders HTTP/1.1" 200 -

```

La commande est bien mise à jour.

Question 5

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