



POLYTECH<sup>®</sup>  
NICE SOPHIA

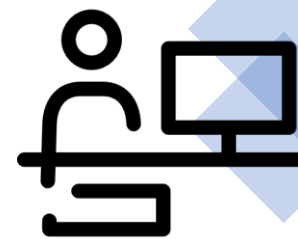


# Livraison des colis par drones

Résilience, Sécurité, Robustesse

ABAKKALI Dina - MARSILI Sylvain - MARTIN Thomas - POUEYTO Clément - STRIEBEL Florian

# Le scope initial de notre sujet



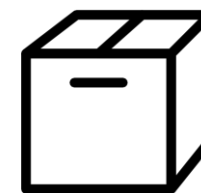
**Commande effectuée**

Livraison de colis aux clients suivant 3 axes:

- Résilience
  - Micro-services et mise en place de Kafka avec 3 brokers
- Sécurité :
  - SSL/SASL dans Kafka
- Robustesse
  - Détection des mauvaises positions



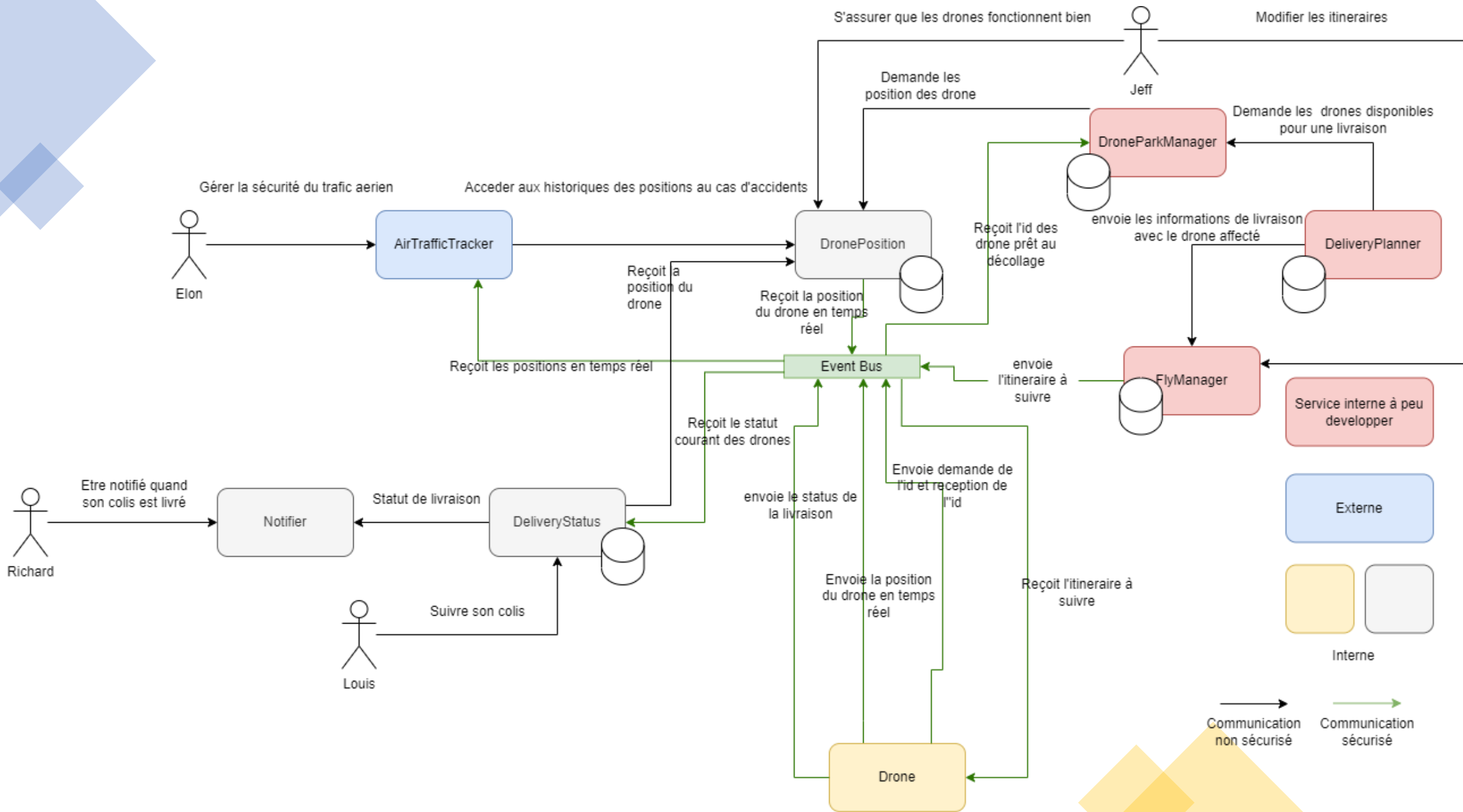
**Livraison lancée**



**Colis livré**



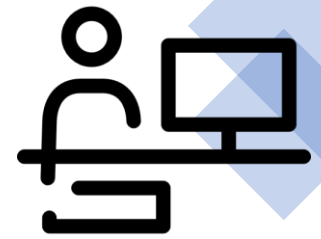
# Architecture initiale



# Le scope actuel de notre sujet

Livraison de colis aux clients suivant 4 axes:

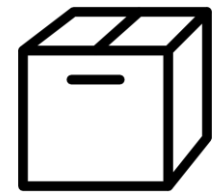
- Charge :
  - un système qui résiste à un grand nombre de drones
- Sécurité :
  - un système qui résiste à une attaque DDoS
- Résilience
  - un démonstrateur qui applique les patterns de résilience sur nos services
  - et des monkeys de démonstration
- Robustesse
  - un système robuste de détection des services défaillants et de redémarrage automatisé



**Commande effectuée**



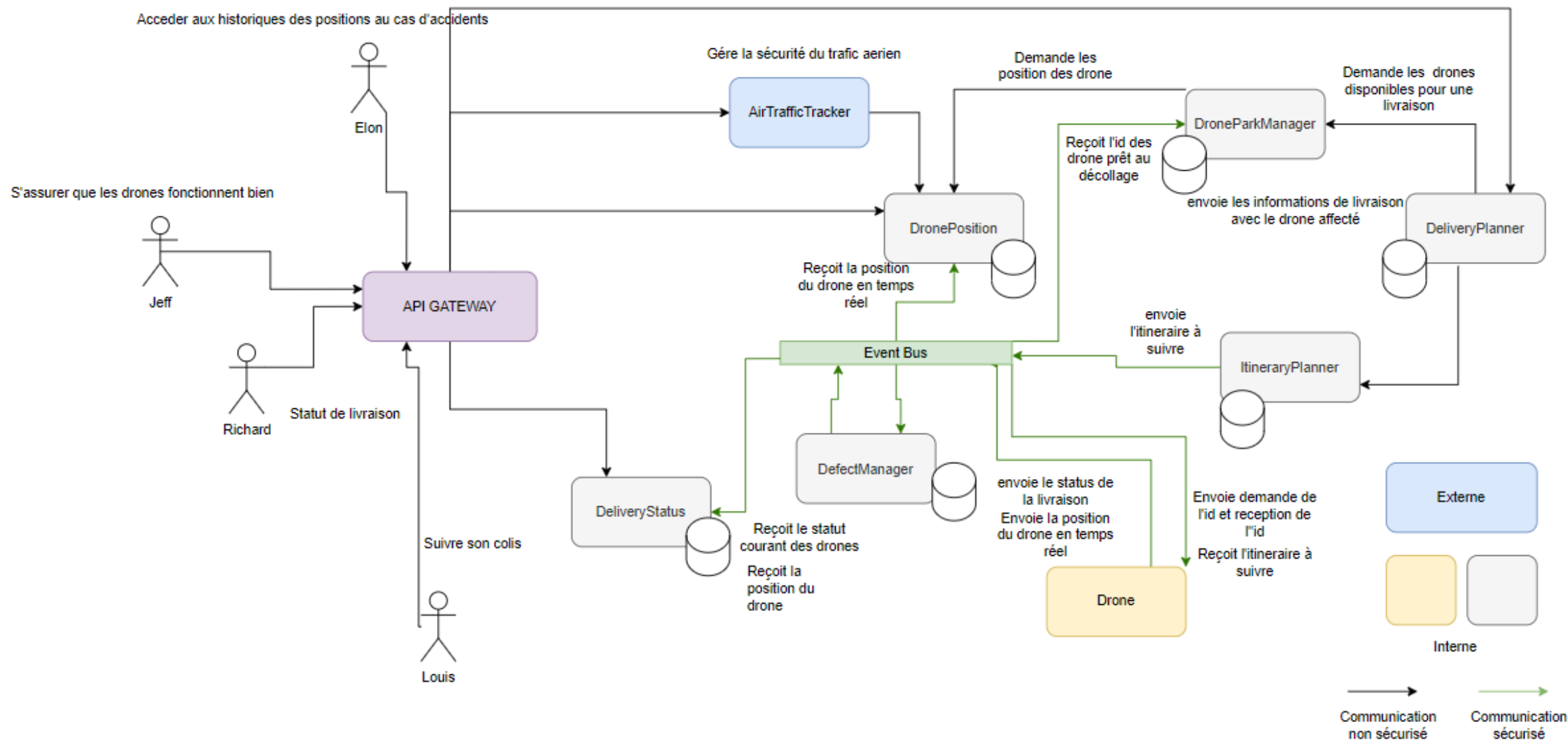
**Livraison lancée**



**Colis livré**



# Architecture actuelle





# Choix d'architecture - Sécurité



# Détecter les attaques DDOS

1 règle snort



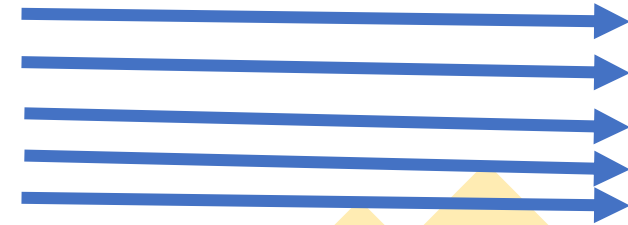
**Alertes**

< 60000 segments tcp en 10 s

1000 drones




5 segments TCP / requête





# Mitiger les attaques DDOS

- Effectuer du filtrage basé sur la position géographique des sources.
  - Bloquer les adresses IP source identifiées comme étant à l'origine de l'attaque.
- 



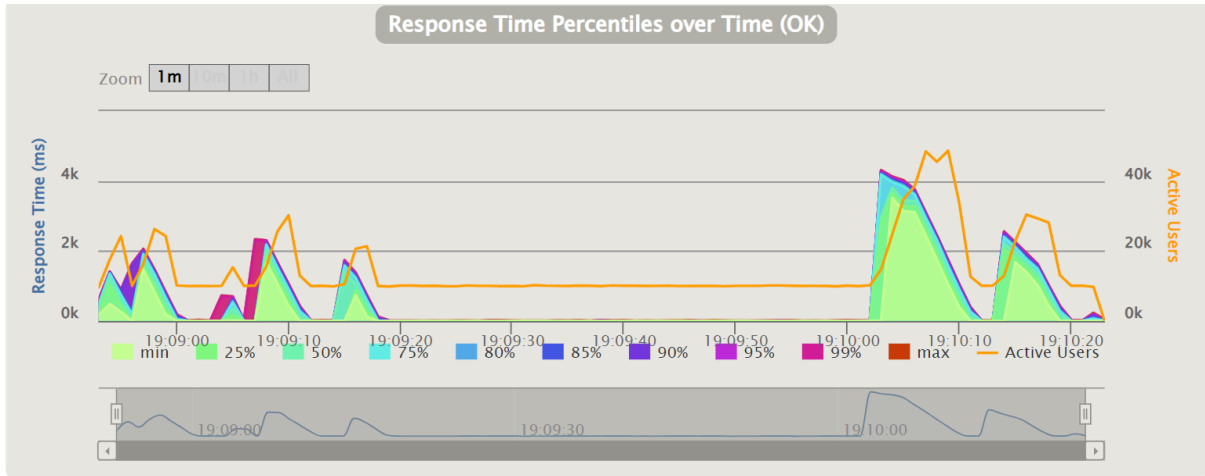
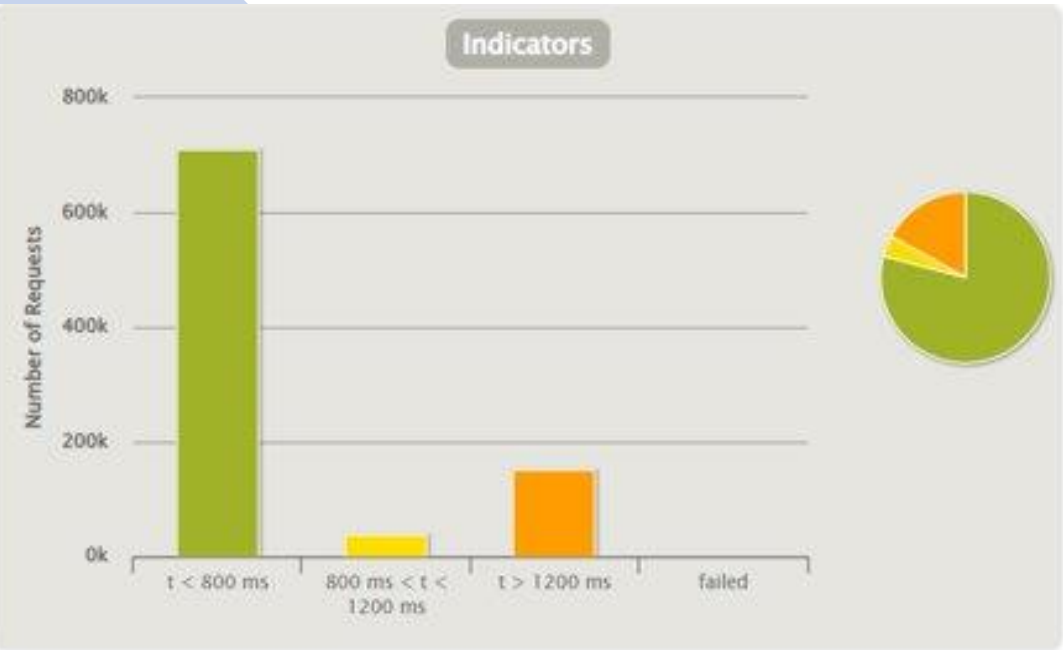
# Choix d'architecture - Charge

# Résister à un grand nombre de drone



- Trouver la limite de charge de l'architecture:
  - Tester l'architecture en utilisation réelle
  - Augmentation du nombre de requêtes à chaque nouvel essaie
- Trouver les points de faiblesses de l'architecture:
  - Les services qui ont eu des erreurs durant l'execution ?
  - Leurs arrêts en a t'il provoqué d'autres ?

tenus



serverDB.positionchangeds

DOCUMENTS 0 STORAGE SIZE 4.1KB AVG. SIZE 0B INDEXES 1 TOTAL SIZE 4.1KB AVG. SIZE 4.1KB

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER { field: 'value' } OPTIONS FIND RESET ↺ ...

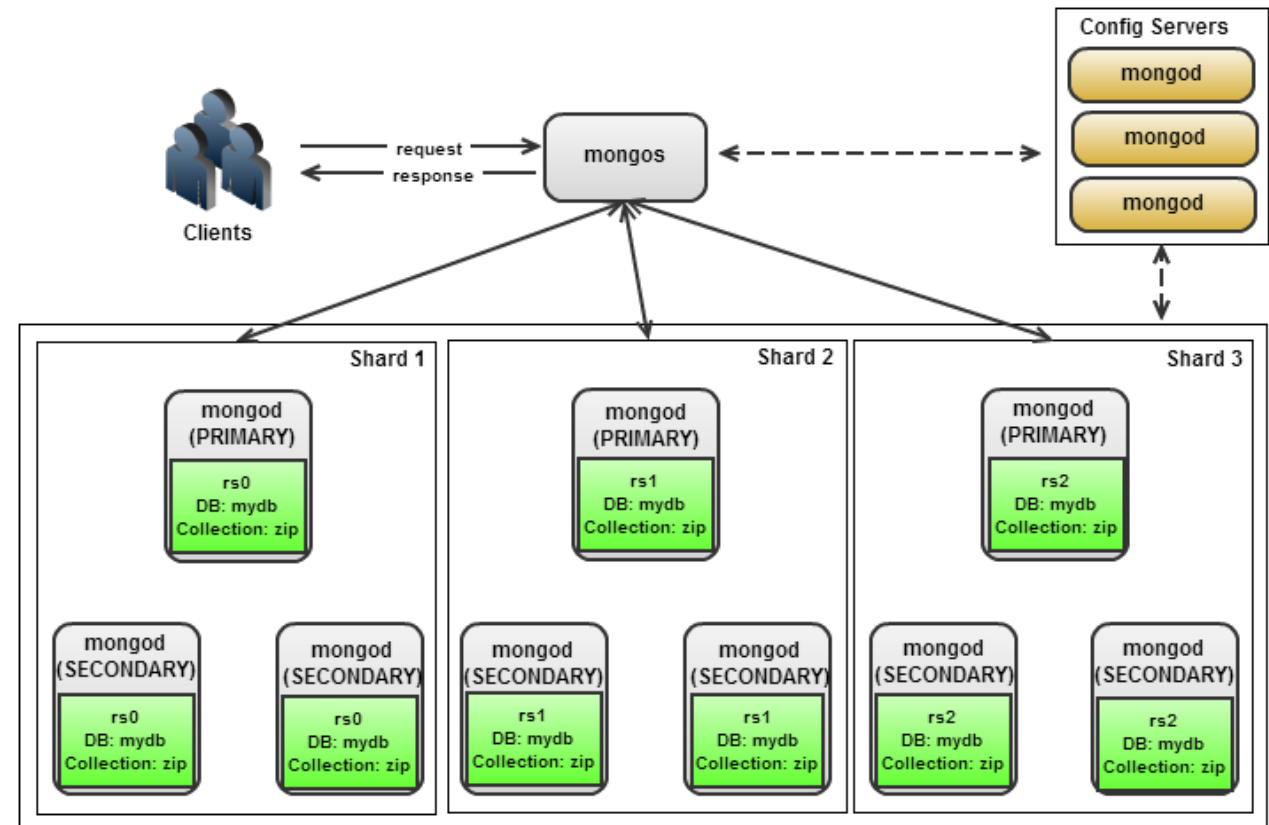
ADD DATA VIEW [ ] [ ] [ ]

Displaying documents - 20 of 36987 REFRESH

```
{
  "_id": ObjectId("6217c945dcf393bcFd7236cc"),
  "position": Object,
  "timestamp": 2022-02-24T07:06:59.000+00:00,
  "droneId": 1,
  "__v": 0
}
```

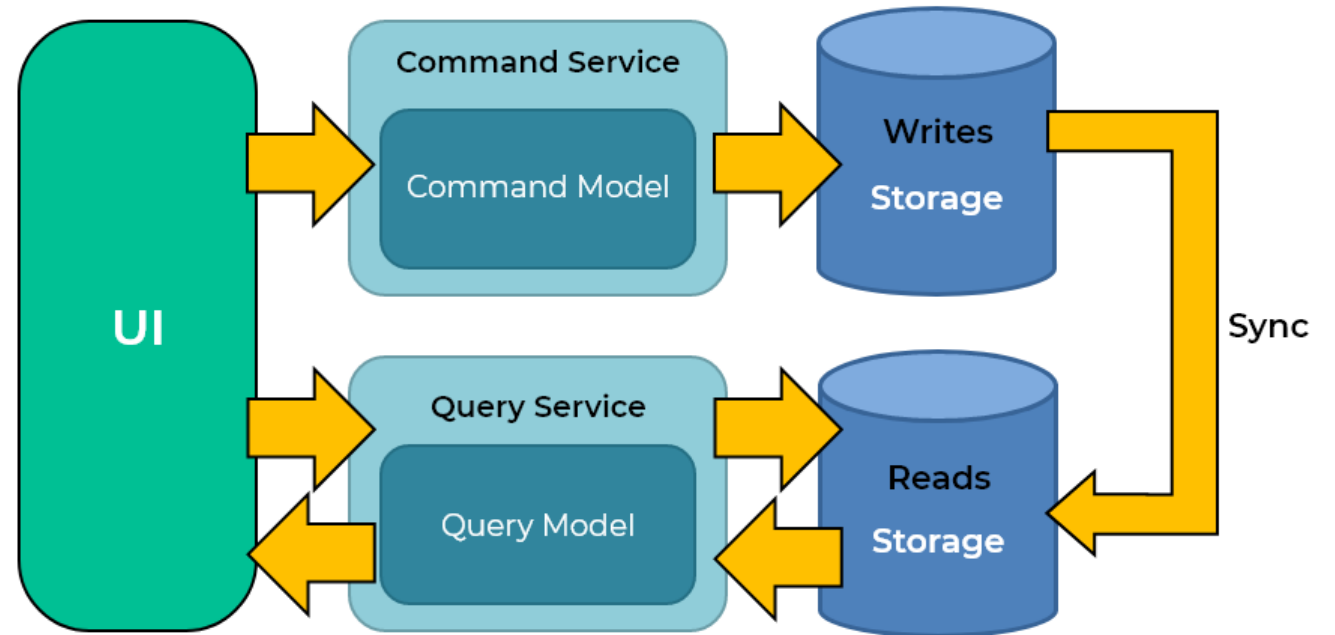
# Résoudre les problèmes de stockage

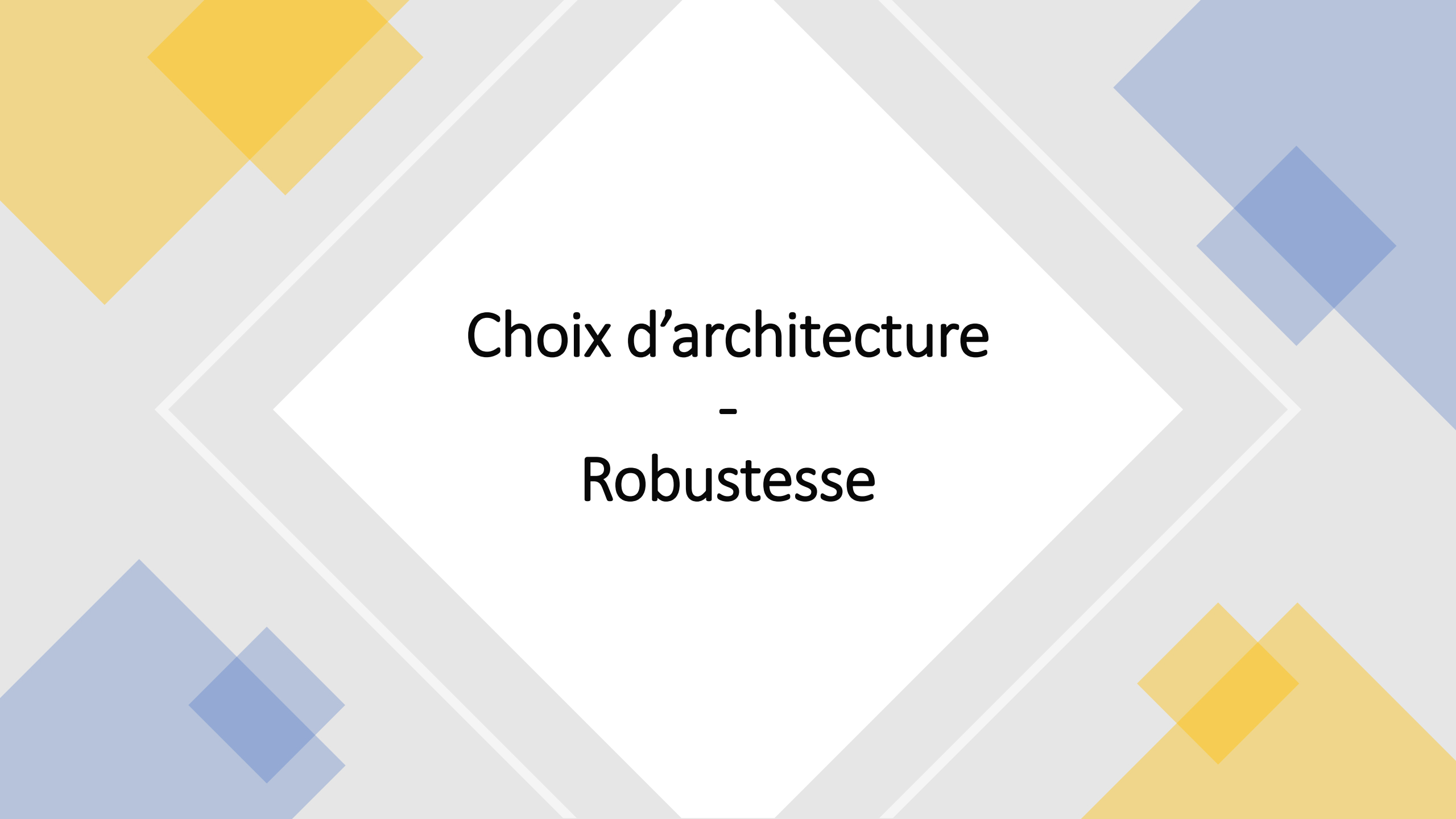
- Faire un cluster MongoDB
  - N'apporte pas de meilleur performance de stockage



# Les autres solutions possibles

- Changer la base de données
- Utilisation du pattern CQRS

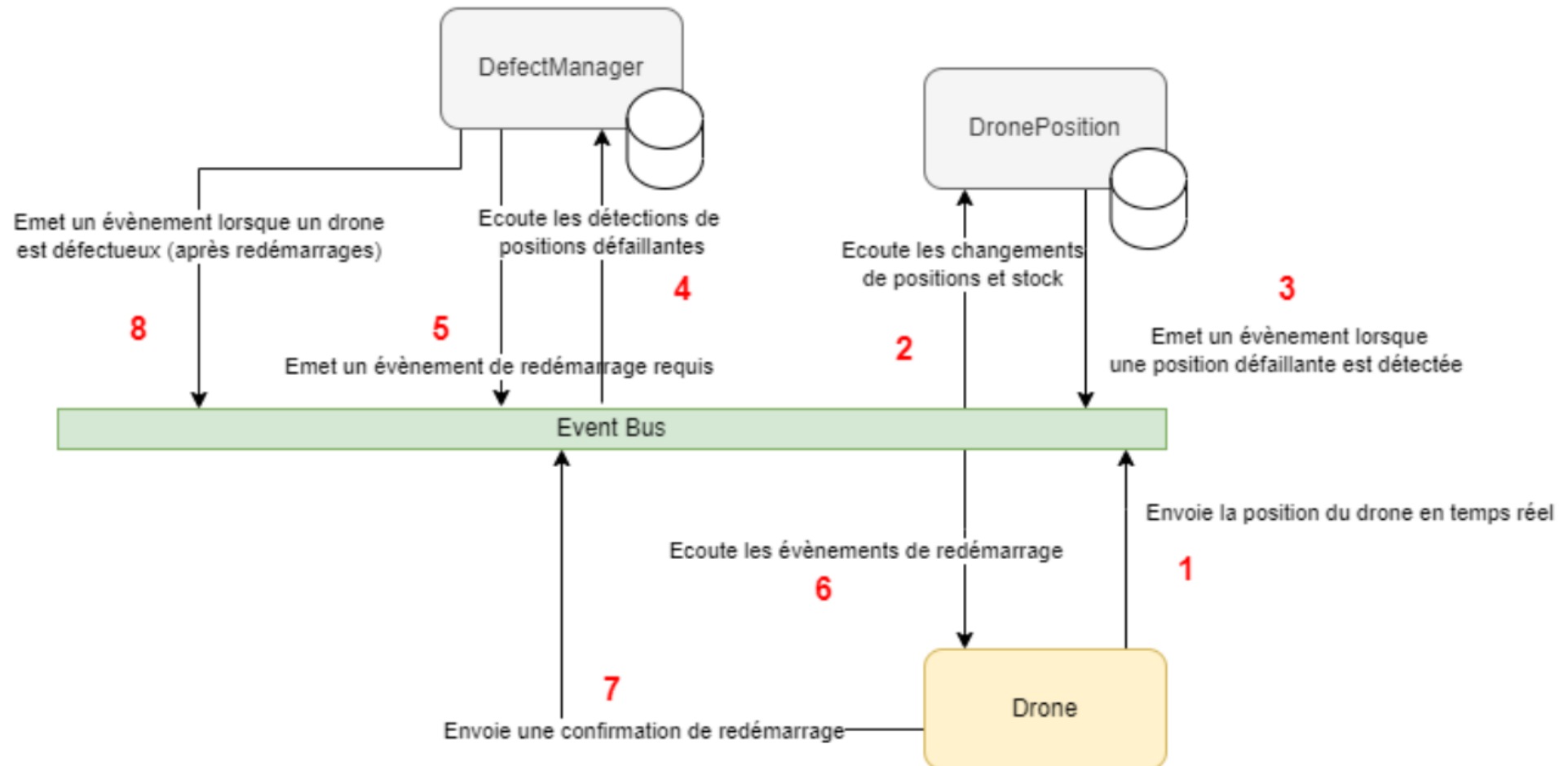




# Choix d'architecture - Robustesse



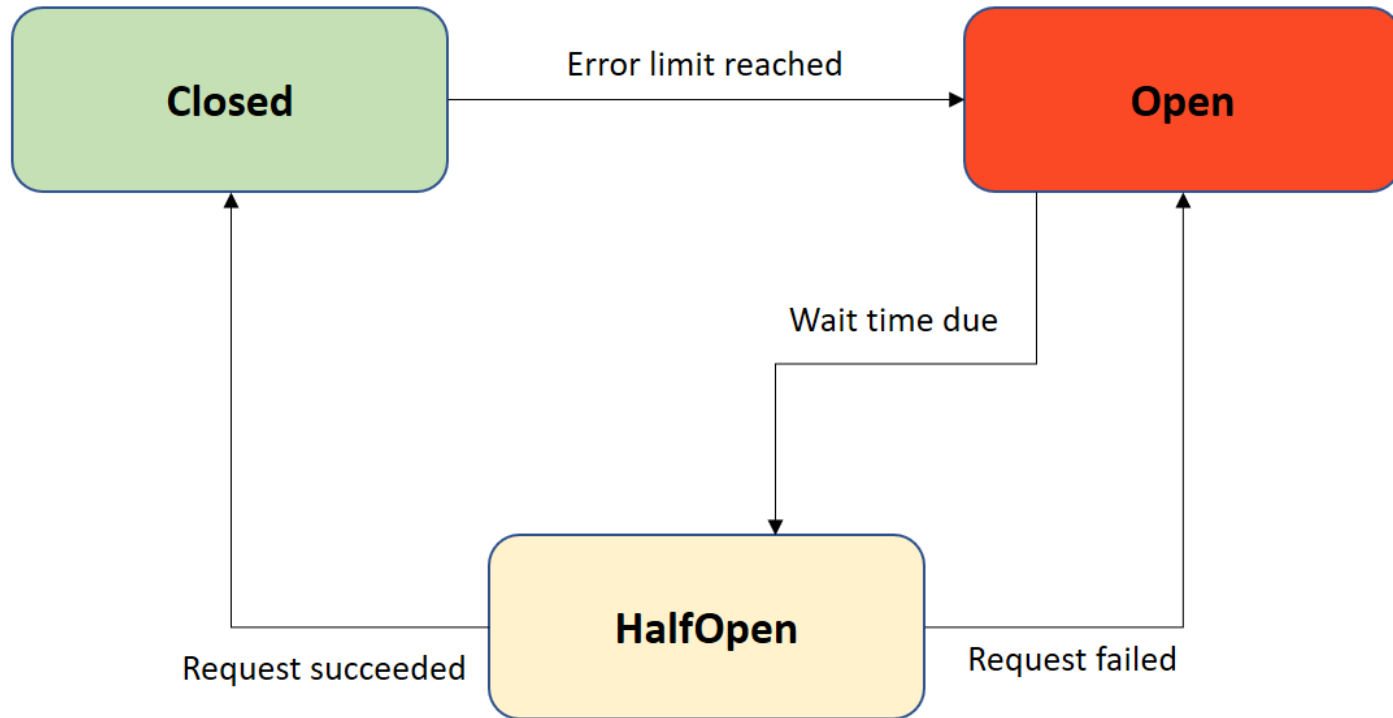
## Détection des mauvaises positions (la suite)





# Choix d'architecture - Résilience

# Utilisation du pattern Circuit Breaker



# Utilisation du pattern Retry





# Choix d'architecture - Déploiement

# Utilisation de docker Swarm

## Robustesse

- Détecte le crash d'un service et en crée un nouveau

## Résilience/Charge

- Définition des réplicas
- Load-balancing
- Répartition sur plusieurs nodes

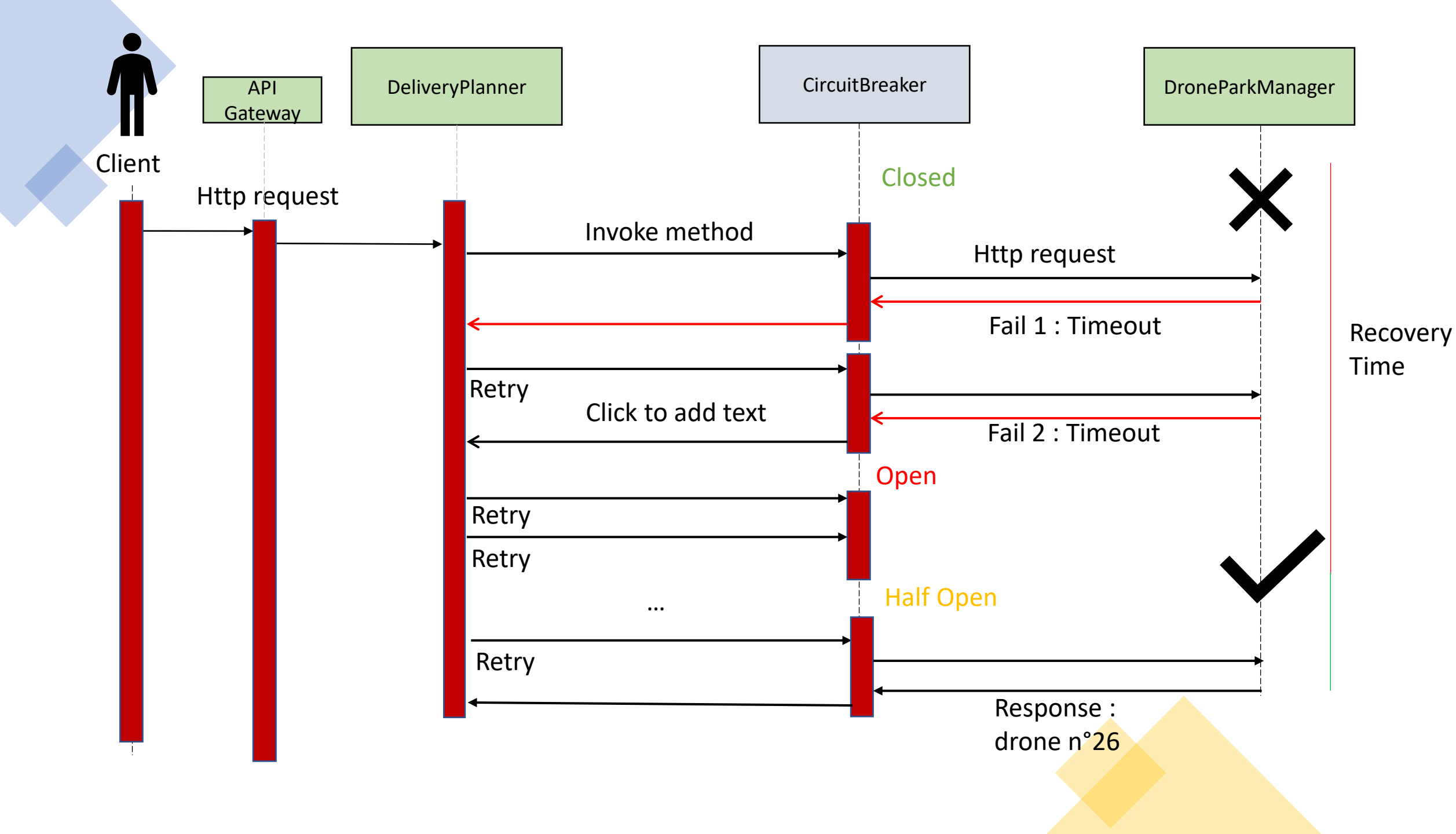





# Démonstrations



# Démonstration - Scénario Résilience







# Démonstration - Scenario 2

# Perspectives futures

- Aller plus loin dans la gestion des drones obsolètes
- Mettre en place une des solutions proposées pour gérer la charge en DB
- Etendre la sécurité au niveau côté client

# Organisation de l'équipe

- Robustesse et tests de charge : Thomas
- Mise en place de Swarm : Sylvain
- Mise en place de Gatling et du Cluster MongoDB : Florian
- Mise en place de Snort : Dina
- API Gateway et patterns de résilience : Clément



Merci