

Documentation de l'architecture du laboratoire de LOG430

- [Documentation de l'architecture du laboratoire de LOG430](#)
- [Page titre](#)
- [Introduction](#)
- [Scénario d'objectif d'affaire](#)
 - [OA-1. Faciliter le recrutement des nouveaux chargés de laboratoire.](#)
 - [OA-2. Validez si le transport par autobus est toujours plus rapide, peu importe l'heure de la journée](#)
- [Cas d'utilisations](#)
 - **CU01 - Veux comparer les temps de trajet.**
 - [CU01-D1 Disponibilité](#)
 - [CU01-M1 Modifiabilité](#)
 - [CU01-P1 Performance](#)
 - [CU01-S1 Sécurité](#)
 - [CU01-T1 Testabilité](#)
 - [CU01-U1 Usabilité](#)
 - [CU01-I1 Interopérabilité](#)
 - **CU02 - Veux pouvoir mettre le chaos dans les microservices.**
 - [CU02-D1 Disponibilité](#)
 - [CU02-M1 Modifiabilité](#)
 - [CU02-P1 Performance](#)
 - [CU02-S1 Sécurité](#)
 - [CU02-T1 Testabilité](#)
 - [CU02-U1 Usabilité](#)
 - [CU02-I1 Interopérabilité](#)
 - **CU03 - Pas de CU03**
 - **CU04 - Veux pouvoir s'authentifier.**
 - [CU04-D1 Disponibilité](#)
 - [CU04-M1 Modifiabilité](#)
 - [CU04-P1 Performance](#)
 - [CU04-S1 Sécurité](#)
 - [CU04-T1 Testabilité](#)
 - [CU04-U1 Usabilité](#)
 - [CU04-I1 Interopérabilité](#)
 - **CU05 - Avoir une intercommunication entre microservices à l'aide d'une source unique de découverte de route.**
 - [CU05-D1 Disponibilité](#)
 - [CU05-M1 Modifiabilité](#)
 - [CU05-P1 Performance](#)

- CU05-S1 Sécurité
- CU05-T1 Testabilité
- CU05-U1 Usabilité
- CU05-I1 Interopérabilité
- CU06 - Veux informer le mainteneur sur le status de vie des microservices
 - CU06-D1 Disponibilité
 - CU06-M1 Modifiabilité
 - CU06-P1 Performance
 - CU06-S1 Sécurité
 - CU06-T1 Testabilité
 - CU06-U1 Usabilité
 - CU06-I1 Interopérabilité
- CU07 - Veux pouvoir informer le mainteneur sur l'état interne d'un service (exemple l'état du CPU)
 - CU07-D1 Disponibilité
 - CU07-M1 Modifiabilité
 - CU07-P1 Performance
 - CU07-S1 Sécurité
 - CU07-T1 Testabilité
 - CU07-U1 Usabilité
 - CU07-I1 Interopérabilité
- CU08 - Veux avoir le temps d'un trajet en autobus
 - CU08-D1 Disponibilité
 - CU08-M1 Modifiabilité
 - CU08-P1 Performance
 - CU08-S1 Sécurité
 - CU08-T1 Testabilité
 - CU08-U1 Usabilité
 - CU08-I1 Interopérabilité
- CU09 - Veux avoir le temps de trajet en auto
 - CU09-D1 Disponibilité
 - CU09-M1 Modifiabilité
 - CU09-P1 Performance
 - CU09-S1 Sécurité
 - CU09-T1 Testabilité
 - CU09-U1 Usabilité
 - CU09-I1 Interopérabilité
- CU10 - Pas de CU10
- Vue architecturale de contexte
 - Présentation primaire
 - Catalogue d'éléments
 - Guide de variabilité
 - Services Externes
 - Base de Données pour l'authentification

- Ajout ou Retrait d'un microservice
- Raisonement
 - Architecture Microservices
 - Patron Discovery
- Conception axée sur les attributs de qualité
 - ADD-Disponibilité
 - ADD-détection de faute
 - ADD-Préparation et réparation
 - ADD-Réintroduction
 - ADD-Prévention des fautes
 - ADD-Modifiabilité
 - ADD-Réduire la taille des modules
 - ADD-Augmenter la cohésion
 - ADD-Réduire le couplage
 - ADD-Defer binding
 - ADD-Performance
 - ADD-Contrôler la demande en ressources
 - ADD-Gérer les ressources
 - ADD-Sécurité
 - ADD-Détecter les attaques
 - ADD-Résister aux attaques
 - ADD-Réagir aux attaques
 - ADD-Récupérer d'une attaque
 - ADD-Testabilité
 - ADD-Contrôle and observe l'état du système
 - ADD-Limiter la complexité
 - ADD-Usabilité
 - ADD-Supporter l'initiative de l'utilisateur
 - ADD-Supporter l'initiative du système
 - ADD-Interopérabilité
 - ADD-Localiser
 - ADD-Gérer les interfaces
- Réalisation des cas d'utilisation
 - **RDCU-CU01** - Veux comparer les temps de trajet.
 - **RDCU-CU02** - Veux pouvoir mettre le chaos dans les services en mode.
 - **RDCU-CU03**
 - **RDCU-CU04** - Veux pouvoir s'authentifier
 - **RDCU-CU05** -
 - **RDCU-CU06** -
 - **RDCU-CU07** -
 - **RDCU-CU08** -
 - **RDCU-CU09** -
 - **RDCU-CU10** -

- **Réalisation des attributs de qualité**
 - **RDAQ-Disponibilité**
 - RDTQ-Détection de faute
 - RDTQ-Préparation et réparation
 - RDTQ-Réintroduction
 - RDTQ-Prévention des fautes
 - Relation entre les éléments architecturaux et les exigences de disponibilité
 - **RDAQ-Modifiabilité**
 - RDTQ-Réduire la taille des modules
 - RDTQ-Augmenter la cohésion
 - RDTQ-Réduire le couplage
 - RDTQ-Defer binding
 - Relation entre les éléments architecturaux et les exigences de disponibilité
 - **RDAQ-Performance**
 - RDTQ-Contrôler la demande en ressources
 - RDTQ-Gérer les ressources
 - Relation entre les éléments architecturaux et les exigences de performance
 - **RDAQ-Sécurité**
 - RDTQ-Détecter les attaques
 - RDTQ-Résister aux attaques
 - RDTQ-Réagir aux attaques
 - RDTQ-Récupérer d'une attaque
 - Relation entre les éléments architecturaux et les exigences de sécurité
 - **RDAQ-Testabilité**
 - RDTQ-Contrôle et observe l'état du système
 - RDTQ-limiter la complexité
 - Relation entre les éléments architecturaux et les exigences de testabilité
 - **RDAQ-Usabilité**
 - RDTQ-Supporter l'initiative de l'utilisateur
 - RDTQ-Supporter l'initiative du système
 - Relation entre les éléments architecturaux et les exigences d'usabilité
 - **RDAQ-Interopérabilité**
 - RDTQ-Localiser
 - RDTQ-Gérer les interfaces
 - Relation entre les éléments architecturaux et les exigences d'interopérabilité
- **Vues architecturales**
 - **Vues architecturales de type Module**
 - Vue #1
 - **Vues architecturales de type composant et connecteur**
 - Vue #1
 - **Vues architecturales de type allocation**
 - Vue #1 - Déploiement pour authentification
- **Conclusion**

- [Documentation des interfaces](#)

Page titre

Introduction

Un service offert aux utilisateurs comporte souvent plusieurs microservices qui travaillent ensemble. Il est difficile de visualiser la complexité que cela peut engendrer en ce qui a trait à l'implémentation. Chaque système est unique et doit atteindre des objectifs de qualité différents. C'est pourquoi la bonne architecture d'un projet est cruciale pour la réalisation et la compréhension des clients externes.

Il est important pour un usager du transport en commun de savoir quel moyen de transport est le plus rapide entre la voiture et l'autobus. C'est pourquoi le but de ce laboratoire est de créer un système, fait de plusieurs microservices, qui offre une comparaison entre les deux moyens de transports.

Le laboratoire comporte 3 tâches, soit la réalisation de cas d'utilisation, la rédaction de la documentation architecturale et finalement l'intégration de microservices faits par d'autres équipes. Ces trois parties permettront d'avoir un système complet et fonctionnel qui possède tous les attributs de qualité. La réalisation des CU choisis par l'équipe, soit l'authentification et le chaos monkey, s'est faite à l'aide de Javascript et de Typescript. La rédaction de l'architecture s'est faite tout au long du développement afin de bien refléter les changements du système. L'intégration s'est faite à la toute fin, avec les cas d'utilisation réalisés par les autres équipes.

Scénario d'objectif d'affaire

OA-1. Faciliter le recrutement des nouveaux chargés de laboratoire.

Notre architecture permet la réalisation du scénario qui a pour objectif de faciliter le recrutement de nouveaux chargés grâce à nos choix concernant l'usabilité. Nous avons choisi de séparer notre architecture en petits modules pour tous nos cas d'utilisation, ce qui rends la correction plus facile pour le chargé de laboratoire et lui permet de passer moins de temps à comprendre notre système. Nous croyons aussi que les diagrammes que nous avons réalisés pour chaque vue contribuent à faciliter la compréhension de nos choix d'architecture. Nous pensons que d'autres éléments faciliteraient le recrutement de nouveaux chargés de laboratoire. Notamment, la division du laboratoire en plusieurs étapes de réalisation. Cela éviterait que toutes les équipes ne demandent une correction complète durant les deux mêmes périodes. De plus, l'obligation d'intégrer le service d'autres équipes peut-être difficile à corriger si certaines équipes n'ont pas terminé leur laboratoire, nous croyons donc que d'offrir une alternative lorsque ce scénario se présente serait bénéfique pour recruter d'autres chargés.

OA-2. Validez si le transport par autobus est toujours plus rapide, peu importe l'heure de la journée

Bien que notre équipe n'ai pas eu comme tâche de réaliser la comparaison des trajets, nous croyons que notre intégration de ce microservice facilite la réalisation de cet objectif d'affaire. En effet, les décisions que nous avons prises en réalisant nos microservices ont le potentiel de permettre au système d'être facilement modifié et de modifier l'interaction avec d'autres services. Bien que cette fonction ne soit pas implémentée, nos choix architecturaux concernant la modifiabilité, soit de réduire le couplage et d'augmenter la cohésion, permettraient facilement d'ajouter des fonctions et donc, d'ajouter les éléments nécessaires à cet objectif d'affaire.

Cas d'utilisations

CU01 - Veux comparer les temps de trajet.

Acteurs externe:

- Utilisateur: Veut pouvoir comparer le temps de trajet en automobile versus autobus.

Précondition:

- L'utilisateur est connecté sur le site Web de comparaisons de trajet.
- Le microservice de comparaison temps de trajet est opérationnel.

Évènement déclencheur:

- L'utilisateur veut se rendre à un endroit en particulier et se demande quel est le meilleur trajet en termes de temps.

Scénario

- L'utilisateur sélectionne une intersection de départ et une intersection d'arrivée, ainsi que le taux de rafraîchissement de la prise de mesure.
- L'utilisateur sélectionne le service externe qu'il veut utiliser pour faire la comparaison des temps de trajet avec les données temps réel de la STM.
- Le système affiche un graphique du temps de déplacement et met celui-ci à jour selon le taux de rafraîchissement.

Évènement résultant:

- Le système affiche un graphique des comparatifs de temps de déplacement qui se met à jour selon le taux de rafraîchissement.

Postcondition:

- Le système est en attente d'une nouvelle commande de l'utilisateur

Cas alternatifs:

1. a **Service externe:** Utiliser plusieurs [services externes](#) disponibles pour faire le comparatif.

Attributs de qualité

Documenter l'ensemble des attributs de qualité qui s'appliquent à ce scénario en terme d'objectif et de mesure.

CU01-D1 Disponibilité

- (SC) Détecter les fautes
- (SC) Préparation et réparation

CU01-M1 Modifiabilité

- (SC) Augmenter la cohésion
- (SC) Différer la liaison

CU01-P1 Performance

- (SC) Contrôler la demande de ressources

CU01-S1 Sécurité

- (SC) Résister aux attaques

CU01-T1 Testabilité

- (SC) Limiter la complexité

CU01-U1 Usabilité

- (SC) Supporter l'initiative du système
- (SC) Supporter l'initiative de l'utilisateur

CU01-I1 Interopérabilité

- N/A

Commentaires:**CU02 - Veux pouvoir mettre le chaos dans les microservices.****Acteurs externe:**

- Chargé de laboratoire: Veut pouvoir faire la correction de chaque cas d'utilisation.

Précondition:

- Tous les microservices sont opérationnels

Évènement déclencheur:

- La documentation pour cet attribut est terminée et l'équipe demande au chargé de laboratoire de corriger celle-ci.
- L'intégration est complétée et l'équipe demande au chargé de laboratoire de corriger celle-ci
- L'implémentation est complétée et l'équipe demande au chargé de laboratoire de corriger celle-ci.

Scénario

1. Un mécanisme automatique et aléatoire de perturbation vient modifier l'architecture de votre système et vous devez vous assurer de quand même respecter les exigences client en terme d'attribut de qualité et de fonctionnalité.

Évènement résultant:

- L'architecture de votre système est perturbée par le mécanisme.
- Le système conserve un log des perturbations
- Le système conserve un log de comment le système a réagi pour résoudre le problème.

Postcondition:

- Les mécanismes de traitement des attributs de qualité détectent le problème et modifient automatiquement l'architecture de votre système pour qu'il continue à respecter les exigences client.

Cas alternatifs:

- 1.a La perturbation consiste à détruire un microservice
- 1.b La perturbation consiste à augmenter la latence d'un microservice

Attributs de qualité

Documenter l'ensemble des attributs de qualité qui s'appliquent à ce scénario en terme d'objectif et de mesure.

CU02-D1 Disponibilité

Prévention de fautes

- Augmenter les compétences

CU02-M1 Modifiabilité

N/A

CU02-P1 Performance

N/A

CU02-S1 Sécurité

N/A

CU02-T1 Testabilité

N/A

CU02-U1 Usabilité

Supporter l'initiative de l'utilisateur

- "Aggregate"

Supporter l'initiative du système

- Maintenir le modèle de tâches

CU02-I1 Interopérabilité

Gérer les interfaces

- "Tailor interface"

Localiser

- Découvrir le service

CU03 - Pas de CU03

CU04 - Veux pouvoir s'authentifier.

Acteurs externe:

- **Chargé de laboratoire:** Veut pouvoir faire la correction de chaque cas d'utilisation.

Précondition:

- tous les microservices sont opérationnels.

Évènement déclencheur:

- La documentation pour cet attribut est terminée et l'équipe demande au chargé de laboratoire de corriger celle-ci.
- L'implémentation est complétée et l'équipe demande au chargé de laboratoire de corriger celle-ci.

Scénario

1. Le (chargé de laboratoire) CL crée un compte utilisateur.
2. Le CL navigue vers la page d'authentification
3. Le CL entre le courriel utilisé pour créer le compte dans la fenêtre appropriée.
4. Le CL entre le mot de passe utilisé pour créer le compte dans la fenêtre appropriée.
5. Le CL clique sur le bouton afin de se connecter.
6. Le service s'ouvre.

Évènement résultant:

- Le CL est authentifié en tant qu'utilisateur du service.
- Le système reconnaît les préférences (s'il y a lieu) de l'utilisateur.

Postcondition:

- Le CL peut naviguer en étant connecté.

Cas alternatifs:

- 1.
 - a) Le CL possède déjà un compte, le système rejette la création du compte.
- 6.
 - a) Le courriel et le mot de passe ne correspondent pas à un compte existant, le système rejette l'authentification.

Attributs de qualité**CU04-D1 Disponibilité***Détection de fautes*

- Doit répondre au ping/echo du service de monitoring.

Détection de fautes

- Si la copie passive (warm) ne reçoit pas de "heartbeat" pendant un certain temps, elle devient la copie principale et averti le service discovery.

Réintroduction

- Redémarrer le service qui s'est arrêté et il devient une copie passive (warm).

Prévention de fautes

- Dans le cas d'une perte de connexion avec la base de données, garder les opérations en mémoire et effectuer une synchronisation.

CU04-M1 Modifiabilité*Réduire le couplage*

- Chaque module du système reste de petite taille.

Augmenter la cohésion

- Chaque module du système d'authentification a un rôle défini.

Defer binding

- Utilisation de l'injection de dépendances.

CU04-P1 Performance

Contrôler la demande en ressources

- Utilisation d'une couche de "cache" afin d'éviter de contacter la base de données trop souvent.

CU04-S1 Sécurité

Détecter les attaques

- Conserver les adresses IP reçues précédemment pour les analyser.

Résister aux attaques

- Encrypter l'information du mot de passe des utilisateurs.
- Refuser les requêtes reçues par les adresses IP inconnues lors du login.

Réagir aux attaques

- Refuser l'accès après 3 demandes d'authentification erronées avec un minuteur.

CU04-T1 Testabilité

Contrôle et observe l'état du système

- Utilisation de sources de données abstraites, pour pouvoir facilement injecter des "mocks" avec l'injection de dépendances.

Limiter la complexité

- Utilisation d'injection de dépendances pour bien cibler les responsabilités des modules et pouvoir les tester indépendamment.

CU04-U1 Usabilité

Convivialité

- Ce service doit être intuitif et suivre les normes des pages de connexion des applications en utilisant une adresse courriel et un mot de passe.

CU04-I1 Interopérabilité

Localiser

- Utilisation du service de découverte pour communiquer avec les autres micro services.

CU05 - Avoir une intercommunication entre microservices à l'aide d'une source unique de découverte de route.

Acteurs externe:

- **Chargé de laboratoire** : Devra corriger ce cas d'utilisation.
- **Service d'authentification** : Devra authentifier les requêtes externes.
- **Tout autre microservice** : Seront en mesure de s'enregistrer.
- **Tout autre microservice** : Seront en mesure de faire rediriger leurs requêtes vers le bon microservice.

Précondition:

- Tous les microservices sont opérationnels.

Évènement déclencheur:

- La documentation pour ce cas d'utilisation est terminée et l'équipe demande au chargé de laboratoire de corriger celle-ci.
- L'intégration est complétée et l'équipe demande au chargé de laboratoire de corriger celle-ci.

Scénario

1. Dans Postman, le chargé de laboratoire entre une route qu'il aimerait accéder.
2. La réponse en provenance de la route est renvoyée.

Évènement résultant:

- La réponse en provenance de la route est renvoyée.

Postcondition:

- Le système est en attente d'une nouvelle requête.

Cas alternatifs:

1. Scénario 1
 - a) Le chargé de laboratoire n'a pas attaché un jeton d'authentification à sa requête.
 - b) Le chargée de laboratoire a entré une route qui n'existe pas.
2. Scénario 2
 - a) Le microservice est indisponible.

Attributs de qualité**CU05-D1 Disponibilité**

- **Détection de faute** : Ping/Echo
- **Préparation et réparation** : Rollback

CU05-M1 Modifiabilité

- **Réduire la taille des modules** : Split Module
- **Augmenter la cohésion** : Increase semantic coherence

CU05-P1 Performance

- **Contrôler la demande en ressources** : Manage sampling rate

CU05-S1 Sécurité

- **Résister aux attaques** : Authenticate actors
- **Réagir aux attaques** : Lock computer

CU05-T1 Testabilité

- **Limiter la complexité** : Limit structural complexity

CU05-U1 Usabilité

CU05-I1 Interopérabilité

- **Locate** : Discover service

Commentaires:

CU06 - Veux informer le mainteneur sur le status de vie des microservices

Acteurs externe:

Les autres microservices

Précondition:

- Connaître l'adresse du Discovery Service
- Tous les microservices sont abonnés au microservice "Service Discovery"
- Le microservice de Monitoring connaît la route du microservice "Service Discovery"

Évènement déclencheur:

- Le système doit s'assurer que l'ensemble de ses microservices sont fonctionnels.

Scénario

- 1- Le microservice de Monitoring questionne le microservice "Service Discovery" sur les routes à ping
- 2- Le microservice "Service Discovery" retourne les routes des autres microservices
- 3- Le microservice de Monitoring questionne les microservices à savoir s'ils sont actifs (ping)
- 4- Les microservices répondent (echo) avec un 200 OK

Évènement résultant:

- Les microservices communiquent leur statut de vie

- Connaître quel microservice est en panne

Postcondition:

S'assurer que les microservices sont fonctionnels.

Cas alternatifs:

4a- Un microservice ne répond pas.

5- Le microservice de Monitoring envoie un message d'alerte pour informer tous les microservices qu'un microservice est en panne.

Attributs de qualité**CU06-D1 Disponibilité**

- (SC) Détection des fautes
- (SC) Prévention des fautes
- (SC) Préparation et réparation
- (SC) Réintroduction

CU06-M1 Modifiabilité

- (SC) Réduire la taille des modules
- (SC) Augmenter la cohésion
- (SC) Réduire le couplage

CU06-P1 Performance

- (SC) Contrôler la demande en ressources

CU06-S1 Sécurité

- (SC) Détecter les attaques
- (SC) Résister aux attaques
- (SC) Réagir aux attaques
- (SC) Récupérer d'une attaque

CU06-T1 Testabilité

- (SC) Contrôler et observer l'état du système

CU06-U1 Usabilité

- (SC) Supporter l'initiative du système

CU06-I1 Interopérabilité

- (SC) Gérer les interfaces.

Commentaires:

CU07 - Veux pouvoir informer le mainteneur sur l'état interne d'un service (exemple l'état du CPU)

Acteurs externe:

- Le microservice "Service Discovery"
- Le microservice de Monitoring
- Les machines virtuelles

Précondition:

- Connaitre l'adresse du Discovery Service
- Tous les microservices sont abonnés au microservice "Service Discovery"
- Le microservice "Service Discovery" connaît les routes des machines virtuelles

Évènement déclencheur:

- Après s'être authentifié au système, le microservice de Monitoring vérifie si toute les machines virtuelles des autres microservices sont en bon état

Scénario

1- Après s'être authentifié au système, le microservice de Monitoring va récupérer les adresses IP des autres microservices

2- Le microservice de Monitoring demande aux VMs des statistiques sur leur composantes internes (CPU, GPU, RAM, etc...)

3- Si les statistiques sont normales, alors le microservice envoi un message pour lui indiquer que les machines virtuelles sur lesquelles les autres microservices sont hébergés vont bien et sont prêtes à être exploitées

4- Le système continue alors son bon fonctionnement

Évènement résultant:

- Le système connaît quelle(s) machine(s) virtuelle(s) est/sont dysfonctionnelle(s)
- Le système ne se lancera pas tant que les machines virtuelles des autres microservices ne sont pas fonctionnelles

Postcondition:

- S'assurer que les machines virtuelles et leurs composantes sont fonctionnelles

Cas alternatifs:

3a.1- Les statistiques d'une ou de plusieurs machines virtuelles sont anormales, indiquant un dysfonctionnement

3a.2- Le système indique aux machines virtuelles dysfonctionnelles de redémarrer

Retour à l'étape 2

Attributs de qualité

CU07-D1 Disponibilité

- (SC) Détection des fautes
- (SC) Prévention des fautes
- (SC) Préparation et réparation
- (SC) Réintroduction

CU07-M1 Modifiabilité

- (SC) Réduire la taille d'une module
- (SC) Augmenter la cohésion
- (SC) Réduire le couplage
- (SC) Defer binding

CU07-P1 Performance

- (SC) Contrôler la demande en ressources

CU07-S1 Sécurité

- (SC) Détecter les attaques
- (SC) Résister aux attaques
- (SC) Réagir aux attaques
- (SC) Récupérer d'une attaque

CU07-T1 Testabilité

- (SC) Contrôle et observe l'état du système.

CU07-U1 Usabilité

- Supporter l'initiative du système

CU07-I1 Interopérabilité

- Gérer les interfaces

Commentaires:

CU08 - Veux avoir le temps d'un trajet en autobus

Acteurs externe:

- Utilisateur: Veut pouvoir obtenir le temps d'un trajet en auto

Précondition:

- Les microservices de proxy, d'authentification et d'obtention de trajet sont opérationnels
- Le serveur du service externe est opérationnel.

Évènement déclencheur:

- L'utilisateur veut le temps d'un trajet en auto.

Scénario

1. L'utilisateur sélectionne une adresse de départ et une adresse d'arrivée.
2. L'utilisateur sélectionne le service externe qu'il veut utiliser pour obtenir le temps de trajet.
3. Le système affiche le temps de trajet le plus court.

Évènement résultant:

- Le système affiche les trois temps de trajet les plus courts avec des départs proches, ainsi que leurs heures de départ.

Postcondition:

- Le système est en attente d'une nouvelle commande de l'utilisateur.

Cas alternatifs:

- N/A

Attributs de qualité**CU08-D1 Disponibilité**

- (SC) Détecter les fautes
- (SC) Préparation et réparation

CU08-M1 Modifiabilité

- (SC) Augmenter la cohésion
- (SC) Réduire le couplage
- (SC) Différer la liaison

CU08-P1 Performance

- N/A

CU08-S1 Sécurité

- (SC) Résister aux attaques

CU08-T1 Testabilité

- (SC) Limiter la complexité

CU08-U1 Usabilité

- (SC) Supporter l'initiative système

CU08-I1 Interopérabilité

- N/A

Commentaires:

CU09 - Veux avoir le temps de trajet en auto

Acteurs externe:

- Utilisateur: Veut pouvoir obtenir le temps d'un trajet en auto

Précondition:

- Les microservices de proxy, d'authentification et d'obtention de trajet sont opérationnels
- Le serveur du service externe est opérationnel.

Évènement déclencheur:

- L'utilisateur veut le temps d'un trajet en auto.

Scénario

1. L'utilisateur sélectionne une adresse de départ et une adresse d'arrivée.
2. L'utilisateur sélectionne le service externe qu'il veut utiliser pour obtenir le temps de trajet.
3. Le système affiche le temps de trajet le plus court.

Évènement résultant:

- Le système affiche le temps de trajet le plus court.

Postcondition:

- Le système est en attente d'une nouvelle commande de l'utilisateur.

Cas alternatifs:

- N/A

Attributs de qualité

CU09-D1 Disponibilité

- (SC) Détecter les fautes
- (SC) Préparation et réparation

CU09-M1 Modifiabilité

- (SC) Augmenter la cohésion
- (SC) Réduire le couplage
- (SC) Différer la liaison

CU09-P1 Performance

- N/A

CU09-S1 Sécurité

- (SC) Résister aux attaques

CU09-T1 Testabilité

- (SC) Limiter la complexité

CU09-U1 Usabilité

- (SC) Supporter l'initiative système

CU09-I1 Interopérabilité

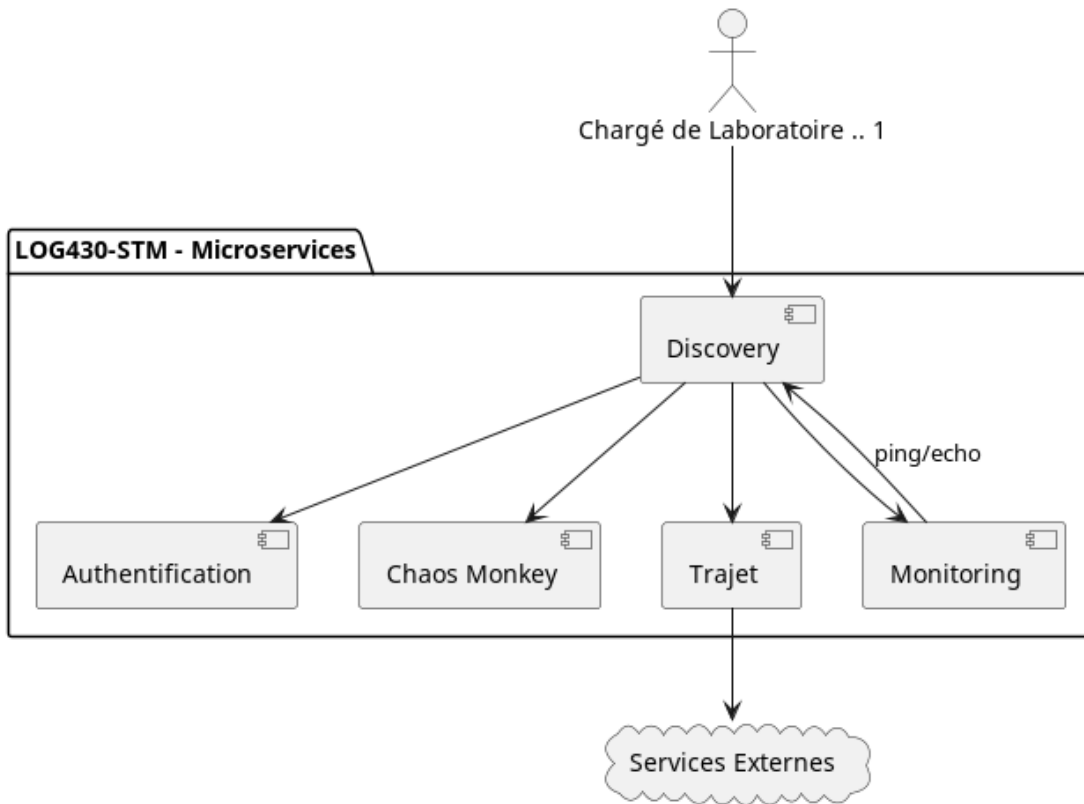
- N/A

Commentaires:

CU10 - Pas de CU10

Vue architecturale de contexte

Présentation primaire



Catalogue d'éléments

Élément	Description	lien vers document d'interfaces
Monitoring	Ce microservice a en charge le monitoring du système en entier, afin de surveiller son bon fonctionnement et d'alerter les autres microservices en cas de panne ou de problème. Les autres microservices doivent s'enregistrer auprès de ce dernier, afin qu'il puisse les ping de temps à autre, en attendant une réponse de ces derniers.	http://www.etsmtl.ca
Chaos Monkey	Ce microservice a été créé pour perturber les autres microservices, en simulant une panne. Il est utilisé pour les tuer, forçant la bonne implémentation du redémarrage ainsi que de s'assurer de la disponibilité du service.	

Élément	Description	lien vers document d'interfaces
Discovery	Le service Discovery est un patron d'architecture utilisé pour simplifier l'interaction entre un élément du système et un autre. Il agit comme une façade, centralisant toutes les interactions entre les différents microservices, ainsi que les acteurs utilisant le système. Le chargé de laboratoire ne pourra accéder aux autres microservices uniquement via ce service, afin de respecter l'esprit du patron.	
Authentification	Ce service d'authentification et d'autorisation sert à vérifier que l'utilisation du système par un utilisateur est sincère est justifié. L'utilisateur concerné doit alors s'authentifier dans le système avant de pouvoir effectuer une action, qui devra être autorisée par ce microservice. Les informations essentielles seront stockées dans une base de données afin d'assurer la persistance et la continuité du service.	Document d'interface Authentification
Trajet	Ce microservice est le point central de nos objectifs d'affaires pour les utilisateurs, Effectuer et comparer un trajet. Ce service contient la logique métier pour réaliser cette opération, en interagissant avec les services externes appropriés afin de récolter les informations ainsi que pour "process" la requête du client et lui retourner les informations voulues.	
Services Externes	Cet élément représente l'ensemble des services avec lesquels le microservice de trajet devra communiquer pour récupérer les informations pertinentes. Il s'agit par exemple des horaires de la STM.	
Chargé de Laboratoire	Le chargé de Laboratoire est le principal utilisateur du système, chargé de vérifier son bon fonctionnement et de corriger les étudiants sur le travail qui a été accompli sur ce système.	

Guide de variabilité

Services Externes

Les services externes sont configurables, divers et variés, ne s'agissant pas d'un seul service centralisé. Il peut s'agir de Google Maps, de la STM, etc. Ces différents services sont interchangeables et leur accès configurable.

Base de Données pour l'authentification

Les paramètres pour l'accès à la base de données contenant les éléments pour l'identification, l'authentification et l'autorisation des utilisateurs est configurable via un fichier de configuration. Ces paramètres prennent la forme d'une URL.

Ajout ou Retrait d'un microservice

Dans le cas du service de Chaos Monkey, du service de monitoring ainsi que du service Discovery, les différents microservices devant être appelés/notifiés/retournés/etc doivent d'abord être enregistrés auprès de ces derniers, via les routes correspondantes.

Raisonnement

Architecture Microservices

Une architecture microservices a été choisie pour une meilleure répartition de la charge de travail entre les différentes équipes. En effet, chaque microservice est indépendant et hautement cohésif, seule une interface commune doit être négociée et documentée pour être partagée avec les autres équipes dans un but commun. Cela permet une meilleure lisibilité du système et une plus grande harmonie, ainsi qu'une répartition équilibrée.

Patron Discovery

Le patron d'architecture Discovery a été choisi afin d'agir comme un Guichet Unique. Dans le cadre d'une architecture microservices, cela a un grand avantage, qui est une meilleure accessibilité pour l'utilisateur, n'ayant qu'un intermédiaire avec qui communiquer, mais aussi pour les autres services : Chaque requête doit être faite via ce microservice avant d'atteindre le service demandé. Cela augmente la clarté pour les équipes de développement, n'ayant qu'un intermédiaire avec qui communiquer, diminue le couplage et augmente le potentiel de scalability du système.

Conception axée sur les attributs de qualité

A partir des qualités associées à tous vos cas d'utilisation, réaliser un mini ADD pour comparer les différents tactiques et identifier clairement la raison de votre choix.

ADD-Disponibilité

Identifiant	Description
CU01-D1	N/A
CU02-D1	Ce service est concerné par la sous-catégorie "Prévention des fautes"

Identifiant	Description
CU03-D1	N/A
CU04-D1	Ce service est concerné par les sous-catégories "Détection de faute", "Préparation et réparation", et "Réintroduction"
CU05-D1	N/A
CU06-D1	N/A
CU07-D1	N/A
CU08-D1	N/A
CU09-D1	N/A
CU10-D1	N/A

ADD-détection de faute

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none">Ping/Echo	<ul style="list-style-type: none">Relativement simple à implémenter	<ul style="list-style-type: none">Nécessite que la copie secondaire connaisse l'adresse de la copie principale	M	M
<ul style="list-style-type: none">Heartbeat	<ul style="list-style-type: none">Relativement simple à implémenter	<ul style="list-style-type: none">Nécessite que la copie principale connaisse l'adresse de la copie secondaire	M	L
<ul style="list-style-type: none">Self-test	<ul style="list-style-type: none">Réduit le nombre de communications inter-processus	<ul style="list-style-type: none">Une autre tactique serait nécessaire pour notifier la copie secondaire que la copie principale n'est plus fonctionnelle	M	H

En combinaison avec les autres tactiques de disponibilité que nous avons décidé d'utiliser, nous avons déterminé que la tactique "Heartbeat" était le plus logique à utiliser. En effet, dans le cadre de la gestion de la copie principale/secondaire, la copie principale sait automatiquement comment communiquer avec la copie secondaire. Un envoi de messages dans cette direction était donc naturel. Si nous avons décidé d'utiliser la tactique "Ping/echo", nous aurions dû également ajouter une configuration initiale permettant à la copie secondaire de communiquer avec la copie principale. Cela aurait augmenté la complexité de la logique de disponibilité. Pour la même raison, utiliser un Self-test aurait été plus compliqué, car il aurait été difficile de notifier la copie secondaire d'un problème au niveau de la copie principale (perte de réseau, crash complet, etc.).

ADD-Préparation et réparation

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Redondance active 	<ul style="list-style-type: none"> Permet de remplacer la copie principale par une copie secondaire 	<ul style="list-style-type: none"> Nécessite que la copie secondaire reçoive tous les messages envoyés à la copie principale Ajoute de la complexité à l'application 	M	H
<ul style="list-style-type: none"> Redondance passive 	<ul style="list-style-type: none"> Permet de remplacer la copie principale par une copie secondaire Dans notre contexte, la copie secondaire n'a pas à se synchroniser avec la copie principale 	<ul style="list-style-type: none"> Ajoute de la complexité à l'application Nécessite de persister à la base de données toutes les opérations 	M	M

Nous avons décidé d'utiliser la tactique de redondance passive, car nous trouvions cette solution plus simple en raison du contexte particulier de notre application. En effet, puisque nous persistons tous les changements à la base de données, aucune synchronisation n'est nécessaire; la copie secondaire est toujours à jour car ses opérations nécessitent de communiquer avec la base de données, qui est à jour. Ainsi, une copie active n'apporterait pas de valeur ajoutée, mais ajouterait encore plus de complexité.

ADD-Réintroduction

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Shadow 	<ul style="list-style-type: none"> Permet de valider le comportement de la copie avant de la rendre disponible au monde extérieur 	<ul style="list-style-type: none"> Augmente la complexité Augmente le temps avant que le composant soit disponible 	M	H
<ul style="list-style-type: none"> State Resynchronization 	<ul style="list-style-type: none"> Permet de mettre à jour la copie afin qu'elle soit prête à recevoir des appels si elle doit devenir la copie principale 	<ul style="list-style-type: none"> Ajoute la tâche de synchroniser la copie secondaire aux responsabilités de la copie principale 	M	M
<ul style="list-style-type: none"> Escalating restart 	<ul style="list-style-type: none"> Permet de rajouter des fonctionnalités à la copie secondaire progressivement 	<ul style="list-style-type: none"> La complexité de l'application ne justifie pas l'ajout progressif de fonctionnalités, puisqu'elle est relativement simple 	L	H

Nous avons choisi d'implémenter la tactique "State Resynchronization" puisque notre stratégie de copies principale/secondaire s'y prêtait bien. En effet, lorsque la copie secondaire devient la copie principale et démarre une nouvelle copie secondaire, celle-ci doit être synchronisée. Si on utilisait les tactiques "Shadow" ou "Escalating restart", cela augmenterait simplement le temps avant que la copie secondaire soit prête, sans augmenter la valeur associée.

ADD-Prévention des fautes

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Transactions 	<ul style="list-style-type: none"> Permet d'éviter des problèmes de corruption de la base de données 	<ul style="list-style-type: none"> Les requêtes envoyées à la base de données sont trop simples pour profiter des transactions 	L	M
<ul style="list-style-type: none"> Exception Prevention 	<ul style="list-style-type: none"> Permet de réduire le nombre d'exceptions, ou de cas d'erreurs, qui apparaissent lors de l'exécution de l'application 	<ul style="list-style-type: none"> L'utilisation de Typescript et Javascript cachent déjà la majorité des exceptions normalement associées à cette tactique 	L	L
<ul style="list-style-type: none"> Increase Competence Set 	<ul style="list-style-type: none"> Permet au logiciel de fonctionner même si d'autres microservices ont des erreurs 	<ul style="list-style-type: none"> Se heurte au théorème de Brewer; la cohérence est sacrifiée au profit de la disponibilité et de la tolérance au partitionnement lors d'une erreur 	H	M

Nous avons choisi d'utiliser la tactique "Increase Competence Set". En effet, si le microservice nous donnant la liste des microservices meurt, ce qui est très probable puisque cette tactique est utilisée dans le cadre du microservice de Chaos, nous devons être en mesure de contrôler les autres microservices en attendant qu'il redémarre.

ADD-Modifiabilité

Identifiant	Description
CU01-M1	N/A
CU02-M1	N/A
CU03-M1	N/A
CU04-M1	Ce service est concerné
CU05-M1	N/A
CU06-M1	N/A
CU07-M1	N/A
CU08-M1	N/A

Identifiant	Description
CU09-M1	N/A
CU10-M1	N/A

ADD-Réduire la taille des modules

Concept de design	Pour	Contre	Valeur	Coût
<ul style="list-style-type: none"> Split Module 	réduit le coût de changement future	N/A	H	H

Nous avons choisi cette tactique parce que si le module en cours de modification comprend de nombreuses fonctionnalités, le coût de la modification sera probablement élevé. Pour remédier à cette situation, nous pouvons affiner le module en plusieurs modules plus petits ce qui devrait réduire le coût moyen des changements futurs.

ADD-Augmenter la cohésion

Concept de design	Pour	Contre	Valeur	Coût
<ul style="list-style-type: none"> Increase Sementic Coherance 	<ul style="list-style-type: none"> Si la responsabilité de A et B dans un module n'a pas le même objectif, elles doivent être placées dans des modules différents 	N/A	H	H

La tactique choisie est *Increase Sementic coherance*. Nous avons choisi cette tactique pour que chaque module possède ses responsabilités ce qui crée un code facile à comprendre et à modifier. De plus, cette tactique garde les dépenses séparées ce qui réduit le couplage et augmente la cohésion.

ADD-Réduire le couplage

Concept de design	Pour	Contre	Valeur	Coût
<ul style="list-style-type: none"> Encapsulate 	<ul style="list-style-type: none"> réduit la probabilité qu'un changement fait dans un module se propage dans un autre module. 	limite les façons dont les responsabilités externes peuvent interagir avec le module	H	M
<ul style="list-style-type: none"> Refactor 	<ul style="list-style-type: none"> Réduis la duplication de code. 	Couteux	M	H
<ul style="list-style-type: none"> Abstract common service 	<ul style="list-style-type: none"> Toute modification du service se produirait en un seul endroit 	Peut devenir très complexe à implémenter	B	M

La tactique choisie est *Encapsulate*. Nous avons choisie cette tactique pour réduire la probabilité qu'un changement fait dans un module se propage dans un autre module.

ADD-Defer binding

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Defer binding 	<ul style="list-style-type: none"> laisser les ordinateurs gérer un changement autant que possible réduira presque toujours le coût de ce changement 	<ul style="list-style-type: none"> Mettre en place les mécanismes pour faciliter cette liaison tardive a tendance à être plus coûteux 	M	M

Nous avons utilisé la tactique de defer binding pour liers des valeurs au moment du déploiement de l'application en utilisant un fichier de ressource (.env) .

ADD-Performance

Identifiant	Description
CU01-P1	N/A
CU02-P1	N/A
CU03-P1	N/A
CU04-P1	Ce service est concerné
CU05-P1	N/A
CU06-P1	N/A
CU07-P1	N/A
CU08-P1	N/A
CU09-P1	N/A
CU10-P1	N/A

ADD-Contrôler la demande en ressources

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Incrémenter l'efficacité des ressources 	Augmenter l'efficacité des algorithmes permettrait une connexion plus rapide lorsque le système recherche les informations nécessaire pour connecter l'utilisateur	Étant donné qu'il y a peu de calculations, il n'y aura pas une grande différence dans la performance	M	M

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Couper le temps d'exécution 	Si la requête d'un utilisateur prends trop de temps à s'effectuer, cela ne bloquera pas d'autres utilisateurs potentiels ayant besoin de l'accès	Un utilisateur pourrait demeurer bloquer et ne jamais pouvoir se connecter	M	M
<ul style="list-style-type: none"> Réduire les coûts d'utilisation 	Utiliser un intermédiaire pour réduire les coûts permet un accès moins fréquent à la base de données et donc un temps de réponse plus rapide	La fidélité de la cache par rapport à la base de données dépend du délai de temps auquel elle est mise à jour, cela pourrait donc conduire à une cache contenant des données qui ne sont plus les bonnes	M	M

Nous avons choisi la tactique "Réduire les coûts d'utilisation" puisque la cache est un intermédiaire qui non seulement augmente la performance, mais conserve aussi une copie de certaines données en cas d'erreur. De plus, c'est un mécanisme qui, s'il est mis à jour fréquemment, est généralement très fiable.

ADD-Gérer les ressources

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Introduire la concurrence 	De la concurrence permettrait à beaucoup de connexions d'être effectuées en même temps, réduisant la charge et l'attente.	Plusieurs fils sont complexes à implémenter dans le cas de l'authentification	M	M
<ul style="list-style-type: none"> Maintenir plusieurs copies des données 	L'utilisation d'un load balancer permettrait de s'assurer que toutes le nombre de requête ne dépasse pas la limite du serveur en utilisant plusieurs serveurs, cela veut dire qu'il y aurait moins de chance qu'une connexion soit refusée par manque de ressources et que les autres connexions se font plus rapidement	L'utilisation de plusieurs serveurs peut s'avérer couteuse et ne s'applique pas à notre laboratoire	M	M
<ul style="list-style-type: none"> tactique 3 	avantages	désavantages	M	M

Nous avons choisi la tactique "xxx"

ADD-Sécurité

Identifiant	Description
CU01-P1	N/A
CU02-P1	N/A
CU03-P1	N/A
CU04-P1	Ce service est concerné.
CU05-P1	N/A
CU06-P1	N/A
CU07-P1	N/A
CU08-P1	N/A
CU09-P1	N/A
CU10-P1	N/A

ADD-Détecter les attaques

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Détecter l'intrusion 	Permet d'identifier des motifs récurrents et connaître les bons utilisateurs	Aucun dans ce contexte	M	M
<ul style="list-style-type: none"> Vérifier l'intégrité du message 	Un très petit changement sera détecté	Difficile dans le cas de l'authentification	M	M

Nous avons choisi la tactique "Détecter l'intrusion" puisqu'elle est une bonne manière de reconnaître un utilisateur qui n'est pas normal et qu'elle est simple à implémenter en comparant les adresses IP

ADD-Résister aux attaques

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Identifier les acteurs 	L'identification permet de facilement refuser un utilisateur anormal	Pourrait refuser l'accès à un utilisateur légitime	M	M
<ul style="list-style-type: none"> Limiter l'accès 	Permet de résister aux utilisateurs mal intentionnés	Tous les utilisateurs doivent avoir accès au service	M	M

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Encrypter les données 	Permet de garantir que les données des utilisateurs ne sont pas volées	L'encryption n'est pas sans failles	M	M

Nous avons choisi la tactique "Identifier les acteurs" puisque malgré qu'il y ait un risque qu'un utilisateur légitime se voit refuser l'accès, cette technique est sécuritaire et s'applique bien au contexte de l'authentification où les utilisateurs reviennent plus d'une fois.

ADD-Réagir aux attaques

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Barrer l'ordinateur 	Barrer le site assure qu'aucune autre attaque n'ait lieu	plus aucun utilisateur ne peut accéder au service	M	M
<ul style="list-style-type: none"> Révoquer l'accès 	Permet de protéger le service d'une personne qui essaie plusieurs mots de passes	pourrait bloquer un utilisateur légitime qui a oublié son mot de passe	M	M

Nous avons choisi la tactique "Révoquer l'accès" puisqu'un utilisateur qui oublie son mot de passe peut potentiellement créer un autre compte, tandis qu'une attaque sera certainement bloquée.

ADD-Récupérer d'une attaque

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Maintenir une piste d'audits 	La piste d'audits permet de retracer les attaquants	ne change pas le système si des dommages ont été faits	M	M

Nous avons choisi la tactique "Maintenir une piste d'audits"

ADD-Testabilité

Identifiant	Description
CU01-P1	N/A
CU02-P1	N/A
CU03-P1	N/A
CU04-P1	Ce service est concerné.
CU05-P1	N/A

Identifiant	Description
CU06-P1	N/A
CU07-P1	N/A
CU08-P1	N/A
CU09-P1	N/A
CU10-P1	N/A

ADD-Controle and observe l'état du système

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none">Abstract Data Sources	<ul style="list-style-type: none">Isolation des modules importantsEncourage cohésionEncourage indépendance des modulesEncourage meilleur définition des interfaces	<ul style="list-style-type: none">Il faut déterminer quels modules sont concernés. La limite, frontière peut être floue.	H	M
<ul style="list-style-type: none">Specialized Interfaces	<ul style="list-style-type: none">Très utile dans de nombreux scénariosEncourage généralisation et standardisation des modules	<ul style="list-style-type: none">Peu utile dans notre microservice d'authentification pour du test"Bloating" du code	B	B
<ul style="list-style-type: none">Sandbox	<ul style="list-style-type: none">Isolation complèteForte cohésion	<ul style="list-style-type: none">Impossible de tester les modules indépendamment	M	H

Nous avons choisi Abstract Data Sources. Cette tactique a été peu coûteuse à implémenter, car nous avons pensé notre système autour de modules indépendants dès le départ (Separation of Concerns). Il a just fallu déterminer quels modules devraient êtres testés et si cette tactique pouvait être appliquée à ceux ci ou non. De plus, cette tactique rapporte la plus haute valeur ajoutée à note application. Non seulement elle est plus testable, mais également plus modifiable et plus maintenable. Cette tactique a donc la plus haute valeur ajoutée.

ADD-Limiter la complexité

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none">Limit Structural Complexity	<ul style="list-style-type: none">Permet d'accomplir d'autres AQs comme	<ul style="list-style-type: none">Complexe à mettre en oeuvre en général	H	M

Concept de design	Pour	Contre	Valeur	Cout
	Modifiabilité <ul style="list-style-type: none"> Réduit couplage Augmente cohésion 	<ul style="list-style-type: none"> Risque d'entrer en contradiction avec d'autres patterns utilisant le polymorphisme 		
<ul style="list-style-type: none"> Limit Nondeterminism 	<ul style="list-style-type: none"> Augmente prédictabilité du système 	<ul style="list-style-type: none"> Pas applicable dans le cadre d'un service d'authentification/autorisation 	B	M

Nous avons choisi Limit Structural Complexity. Ce choix a été assez rapide, pour 2 raisons principales. La première, la tactique de limiter le nondéterminisme ne s'applique peu, voir pas du tout, dans le cadre d'un microservice d'authentification et d'autorisation, qui est déterministe par définition (un login correspond à un token qui est unique. Il ne peut pas en générer un autre à un même instant T). La deuxième raison est qu'il a été facile d'implémenter cette tactique pour notre système, facilement découplable en modules indépendants, très cohésifs avec peu de couplage.

ADD-Usabilité

Identifiant	Description
CU01-U1	N/A
CU02-U1	Ce service est concerné
CU03-U1	N/A
CU04-U1	N/A
CU05-U1	N/A
CU06-U1	N/A
CU07-U1	N/A
CU08-U1	N/A
CU09-U1	N/A
CU10-U1	N/A

ADD-Supporter l'initiative de l'utilisateur

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Cancel 	<ul style="list-style-type: none"> Permettre à l'utilisateur d'annuler une erreur commise 	<ul style="list-style-type: none"> Ajoute beaucoup de complexité à l'exécution de la requête 	M	H

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Undo 	<ul style="list-style-type: none"> Permettre à l'utilisateur d'annuler une erreur commise 	<ul style="list-style-type: none"> Ajoute beaucoup de complexité à l'exécution de la requête 	M	H
<ul style="list-style-type: none"> Pause/Resume 	<ul style="list-style-type: none"> Permettre à l'utilisateur de valider que sa commande est correcte avant de continuer 	<ul style="list-style-type: none"> Ajoute beaucoup de complexité à l'exécution de la requête 	M	H
<ul style="list-style-type: none"> Aggregate 	<ul style="list-style-type: none"> Permet à l'utilisateur d'effectuer plusieurs actions en même temps, ce qui réduit le nombre de clics nécessaires 	<ul style="list-style-type: none"> Ajoute un peu de complexité à l'exécution de la requête 	H	L

Nous avons choisi d'utiliser la tactique "Aggregate", puisqu'elle est la plus simple à implémenter et que la valeur ajoutée est plus grande. En effet, toutes les autres tactiques se concentrent sur la modification d'une requête après son envoi, ce qui est moins utile dans notre contexte puisque la simplicité des requêtes réduit le risque d'erreurs. Il est ainsi plus utile de pouvoir permettre à l'utilisateur de réduire le nombre de clics pour affecter plusieurs microservices, ce qui réduit également le nombre d'erreurs.

ADD-Supporter l'initiative du système

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Maintain task model 	N/A	Ne s'applique pas	L	M
<ul style="list-style-type: none"> Maintain user model 	N/A	Ne s'applique pas	L	M
<ul style="list-style-type: none"> Maintain system model 	<ul style="list-style-type: none"> Permet à l'utilisateur de connaître l'état de sa requête 	<ul style="list-style-type: none"> Ajoute beaucoup de complexité au système 	M	H

On ne peut pas utiliser la tactique "Maintain task model" puisque la tâche à effectuer par l'utilisateur est beaucoup trop simple pour en bénéficier, et on ne peut pas utiliser la tactique "Maintain user model" pour la même raison. On doit donc utiliser la tactique "Maintain system model", implémentée de manière à pouvoir fournir à l'utilisateur l'état d'une requête. Cette tactique demande cependant beaucoup d'effort à implémenter puisque l'on doit retourner à l'utilisateur une manière de suivre l'état de sa requête avant la complétion de celle-ci, ce qui demande d'ajouter des tâches en arrière-plan.

ADD-Interopérabilité

Identifiant	Description
CU01-I1	N/A
CU02-I1	Ce service est concerné pour toutes les sous-catégories
CU03-I1	N/A
CU04-I1	Ce service est concerné pour la sous-catégorie "Localiser"
CU05-I1	N/A
CU06-I1	N/A
CU07-I1	N/A
CU08-I1	N/A
CU09-I1	N/A
CU10-I1	N/A

ADD-Localiser

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Discover Service 	<ul style="list-style-type: none"> Permet de gérer dynamiquement quels services sont disponibles 	<ul style="list-style-type: none"> Augmente le nombre d'appels qui doivent être faits afin de communiquer avec d'autres microservices 	H	M

Nous avons choisi la tactique "Discover Service" car elle nous permet, dans tous nos microservices, de connaître les autres microservices existants. En effet, surtout dans le cas du microservice de Chaos, il est impératif d'obtenir facilement une liste à jour de tous les microservices déployés.

ADD-Gérer les interfaces

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Orchestrate 	<ul style="list-style-type: none"> Permet de faire collaborer plusieurs services indépendants afin d'effectuer une action plus complexe 	<ul style="list-style-type: none"> Les actions effectuées par nos microservices ne sont pas assez complexes pour nécessiter l'utilisation de plusieurs petits services indépendants. Cela augmenterait beaucoup la complexité et la latence. 	L	H

Concept de design	Pour	Contre	Valeur	Cout
<ul style="list-style-type: none"> Tailor interface 	<ul style="list-style-type: none"> Permet de configurer certaines interfaces afin d'offrir plus ou moins de fonctionnalités selon certains paramètres, ici le type d'utilisateur (normal ou "poweruser") 	<ul style="list-style-type: none"> Augmente la complexité de l'application et la configuration requise pour faire certains appels 	M	M


Nous avons choisi la tactique "Tailor interface" car celle-ci nous permet d'offrir des fonctionnalités plus puissantes à certains utilisateurs, ce qui est utile puisque la nature du microservice de Chaos se prête bien à la présence de différents types d'utilisateur. Dans notre cas, nous avons décidé d'offrir une fonctionnalité de "Batching", ou de faire plusieurs appels en un, aux utilisateurs de type "poweruser".

Réalisation des cas d'utilisation


RDCU-CU01 - Veux comparer les temps de trajet.

 Diagramme

RDCU-CU02 - Veux pouvoir mettre le chaos dans les services en mode.

 DiagrammeSéquenceChaosMonkey

RDCU-CU04 - Veux pouvoir s'authentifier

 DiagrammeSéquenceAuthentification

RDCU-CU05 -

 Diagramme

RDCU-CU06 -

 Diagramme

RDCU-CU07 -

 Diagramme

RDCU-CU08 -

 Diagramme

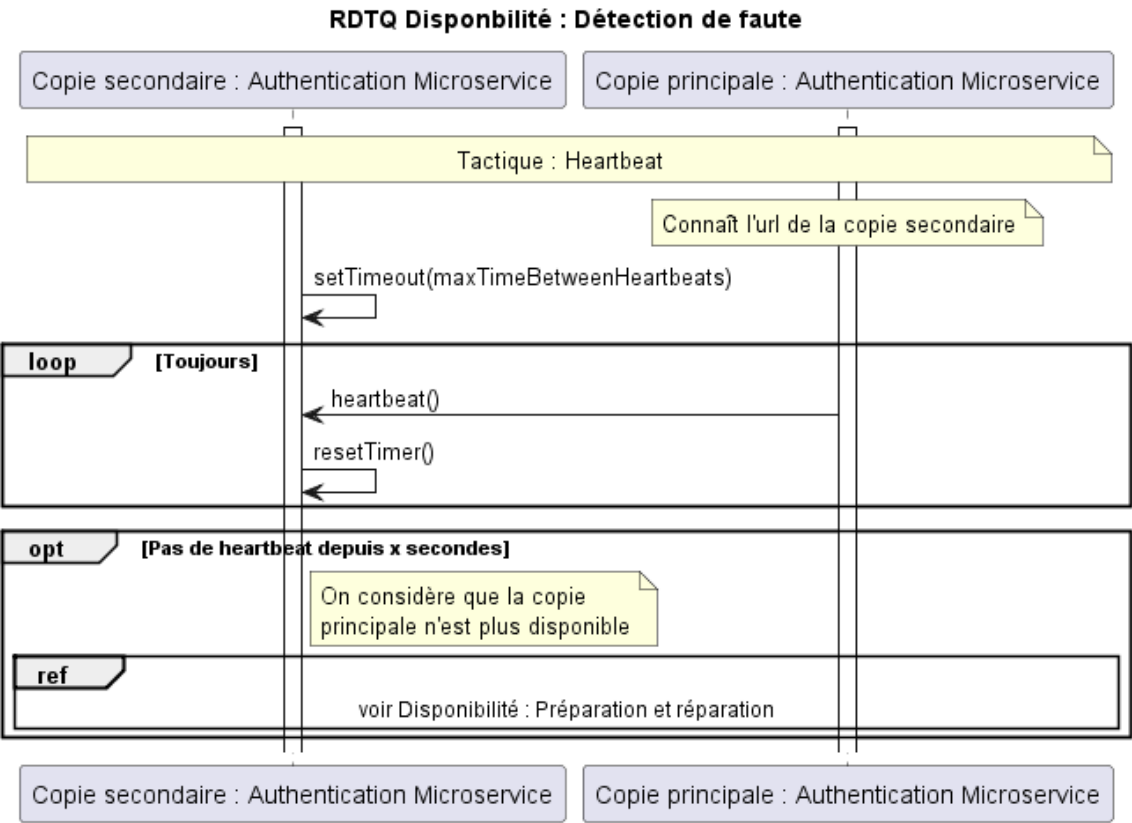
RDCU-CU09 -

 Diagramme

Réalisation des attributs de qualité

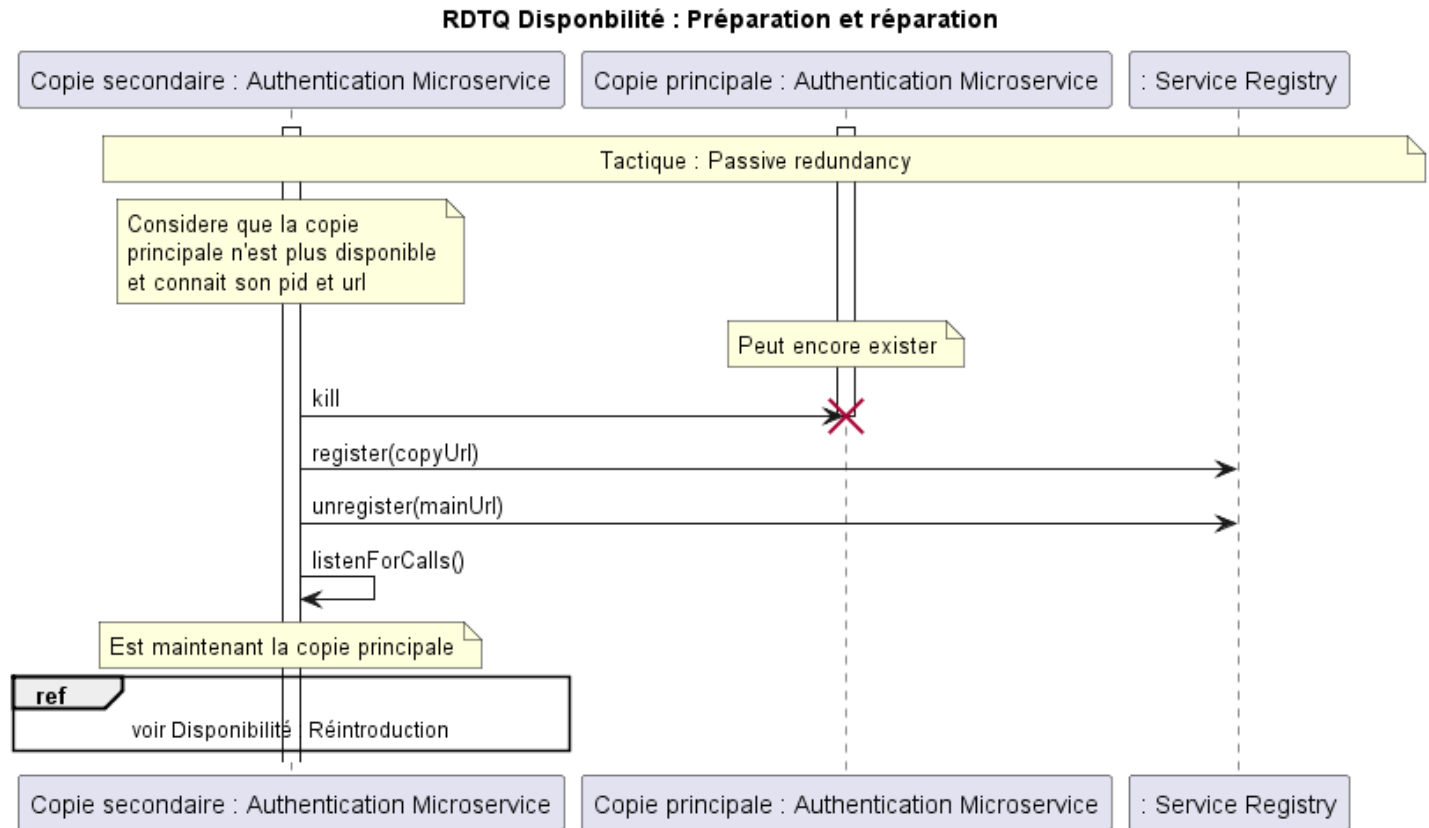
RDAQ-Disponibilité

RDTQ-Détection de faute



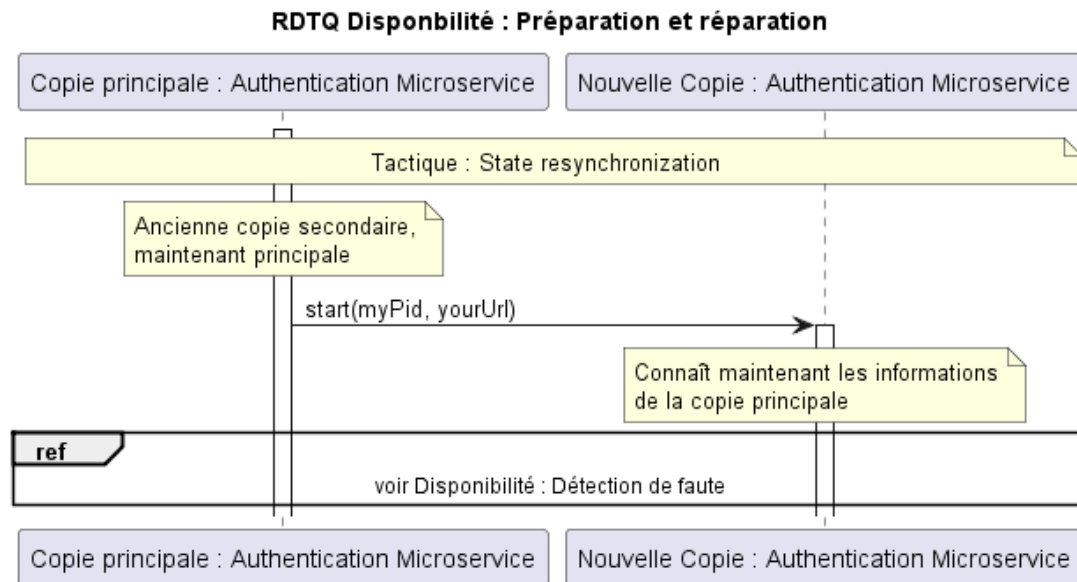
La copie principale envoie des messages périodiques à la copie secondaire, appelés Heartbeats, de manière à lui faire savoir qu'elle est toujours disponible.

RDTQ-Préparation et réparation



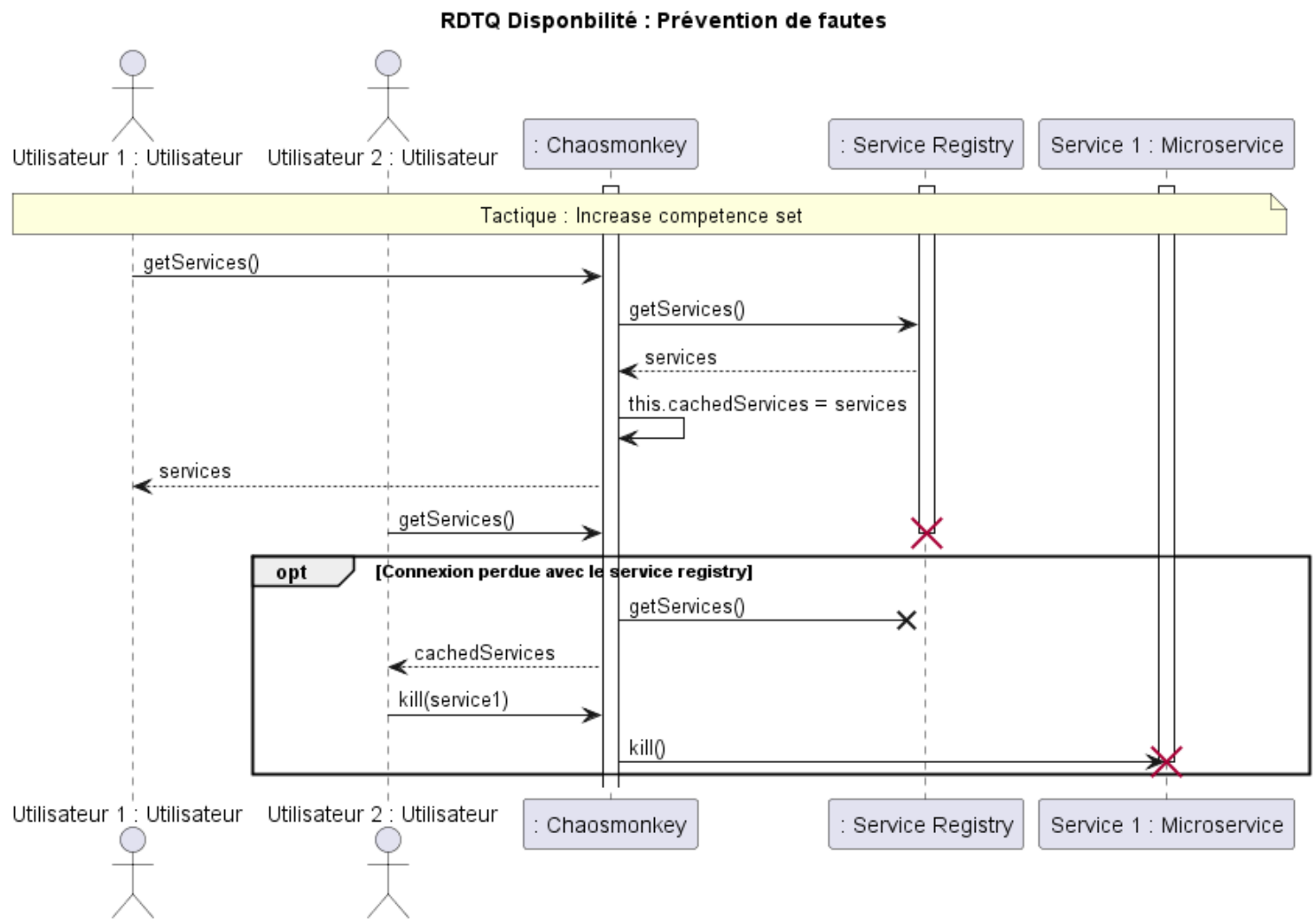
La redondance passive est utilisée afin de toujours avoir une copie de l'application qui est prête à recevoir les appels.

RDTQ-Réintroduction



Si la copie secondaire devient principale, elle doit démarrer une autre copie secondaire afin de maintenir deux copies en tout temps.

RDTQ-Prévention des fautes



Afin de permettre au Chaosmonkey de fournir des services même si le ServiceDiscovery est indisponible, on garde la liste de services en cache.

Relation entre les éléments architecturaux et les exigences de disponibilité

Identifiant	Éléments	Description de la responsabilité
CU02-D1	Chaosmonkey	Doit être disponible même si le Service Registry est hors-ligne
CU02-D2	Service Registry	Doit donner la liste des services
CU04-D1	Copie principale	Doit démarrer la copie secondaire et lui envoyer des heartbeats
CU04-D2	Copie secondaire	Doit s'assurer que la copie principale fonctionne. Sinon, elle devient la copie principale et démarre une nouvelle copie secondaire
CU04-D2	Nouvelle copie secondaire	Doit se synchroniser afin de pouvoir être disponible si la copie secondaire arrête de fonctionner

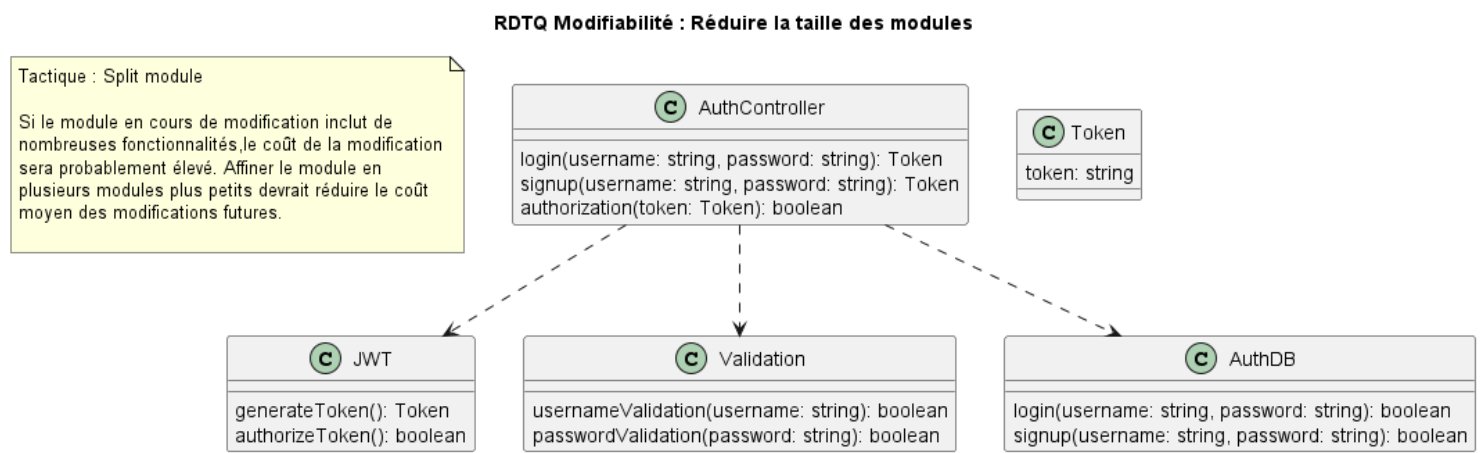
12/8/22, 10:05 AM

documentationArchitecture

Identifiant	Éléments	Description de la responsabilité
CU04-D3	Service Registry	Doit être notifié d'un changement de la copie principale

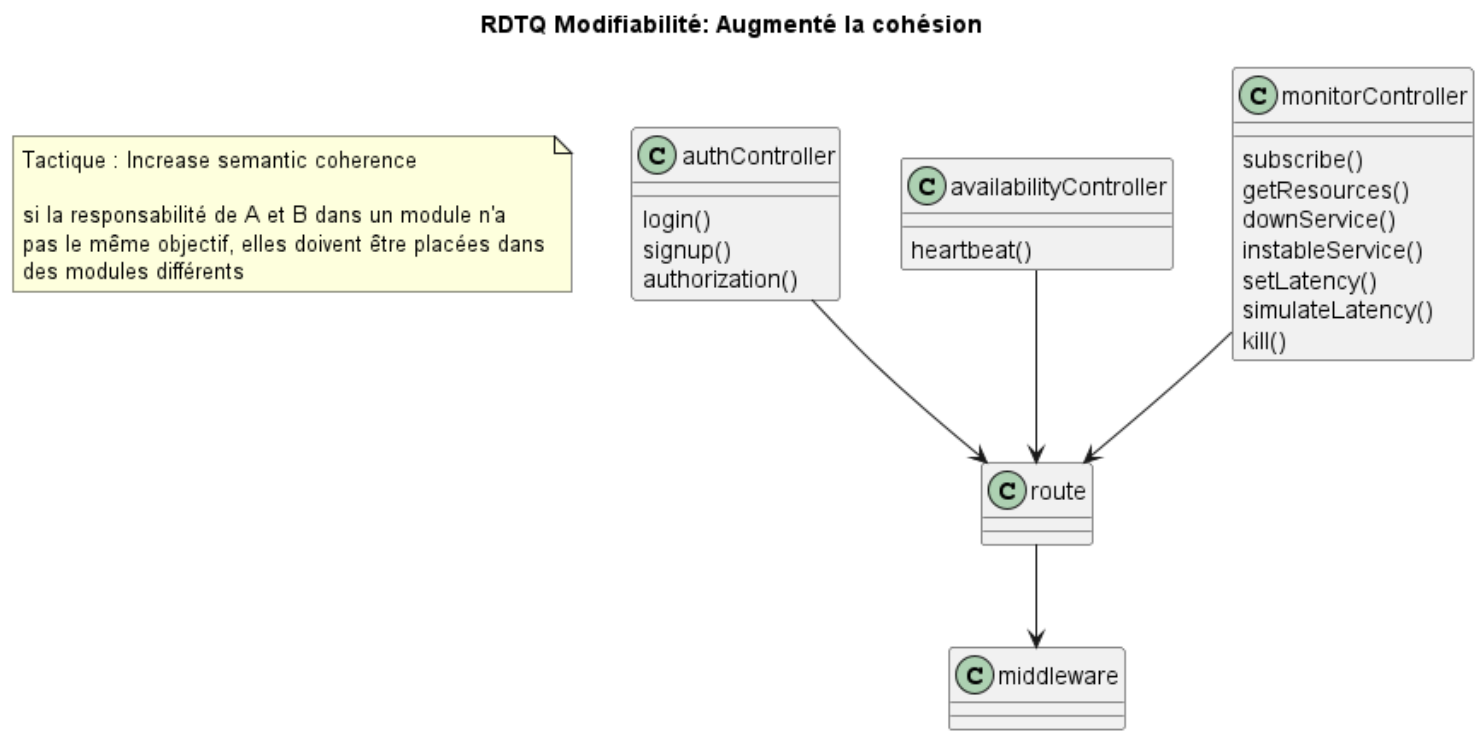
RDAQ-Modifiabilité

RDTQ-Réduire la taille des modules



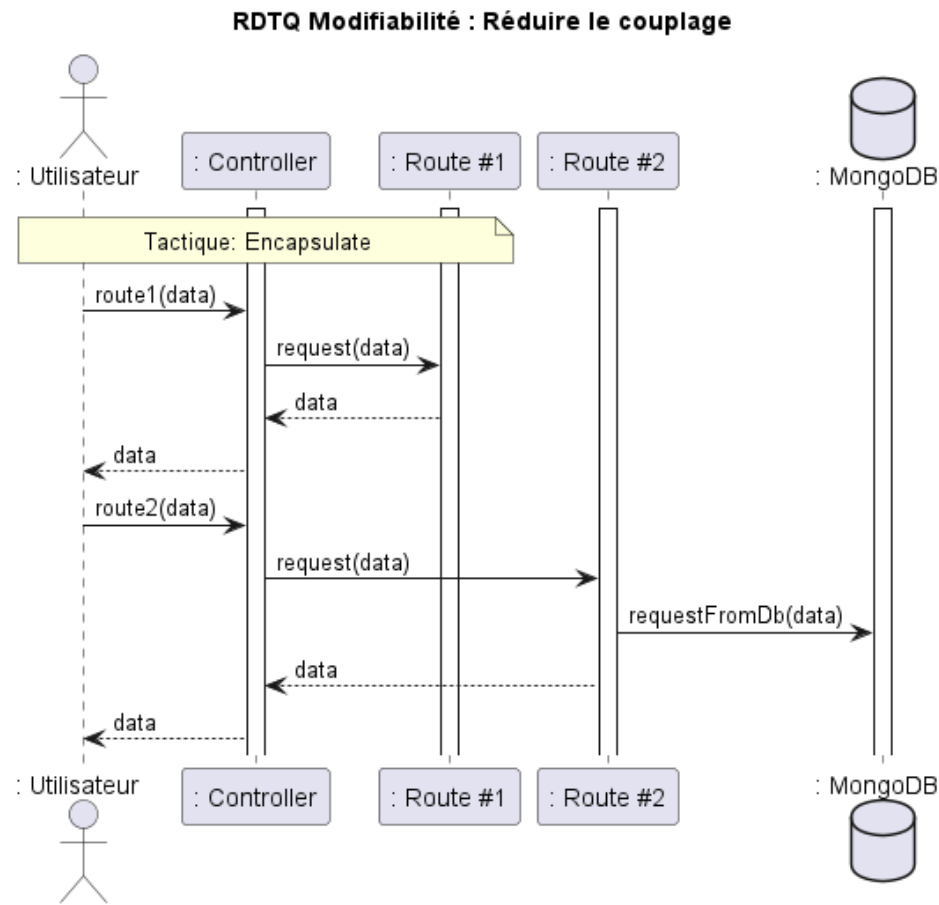
RDTQ-Augmenter la cohésion

Increase semantic coherence

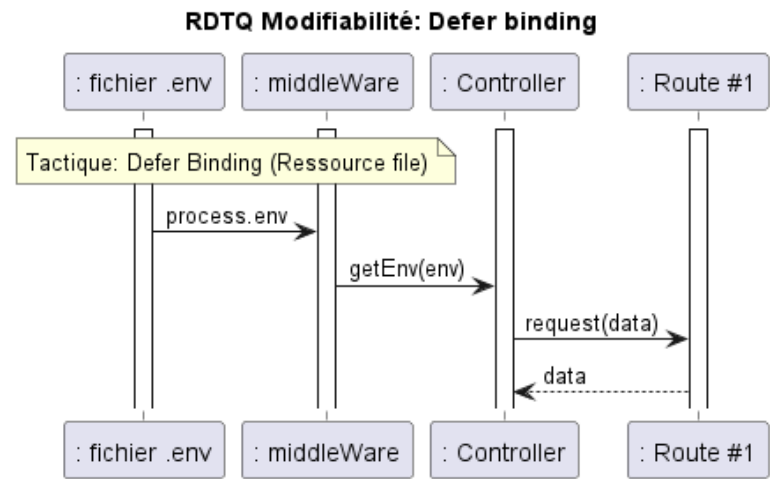


RDTQ-Réduire le couplage

Encapsulation



RDTQ-Defer binding



L'utilisation d'un fichier de configuration permet d'attendre plus longtemps avant d'insérer certaines valeurs dans des variables afin de réduire le couplage.

Relation entre les éléments architecturaux et les exigences de disponibilité

Identifiant	Éléments	Description de la responsabilité
CU04-M1	Fichier .env	Contient des informations qui seront utilisées par l'application au moment de l'exécution

RDAQ-Performance

RDTQ-Contrôler la demande en ressources



ContrôlerLaDemande

RDTQ-Gérer les ressources



GérerLesRessources

Relation entre les éléments architecturaux et les exigences de performance

Identifiant	Éléments	Description de la responsabilité
CU04-P1	Utilisateur	Attends un service d'authentification performant
CU04-P2	JWTService	Fais les gestion des token d'utilisateurs
CU04-P3	Cache	Garde en mémoire rapide les token

RDAQ-Sécurité

RDTQ-Détecter les attaques



DétecterLesAttaques

RDTQ-Résister aux attaques



RésisterAuxAttaques

RDTQ-Réagir aux attaques



RéagirAuxAttaques

RDTQ-Récupérer d'une attaque



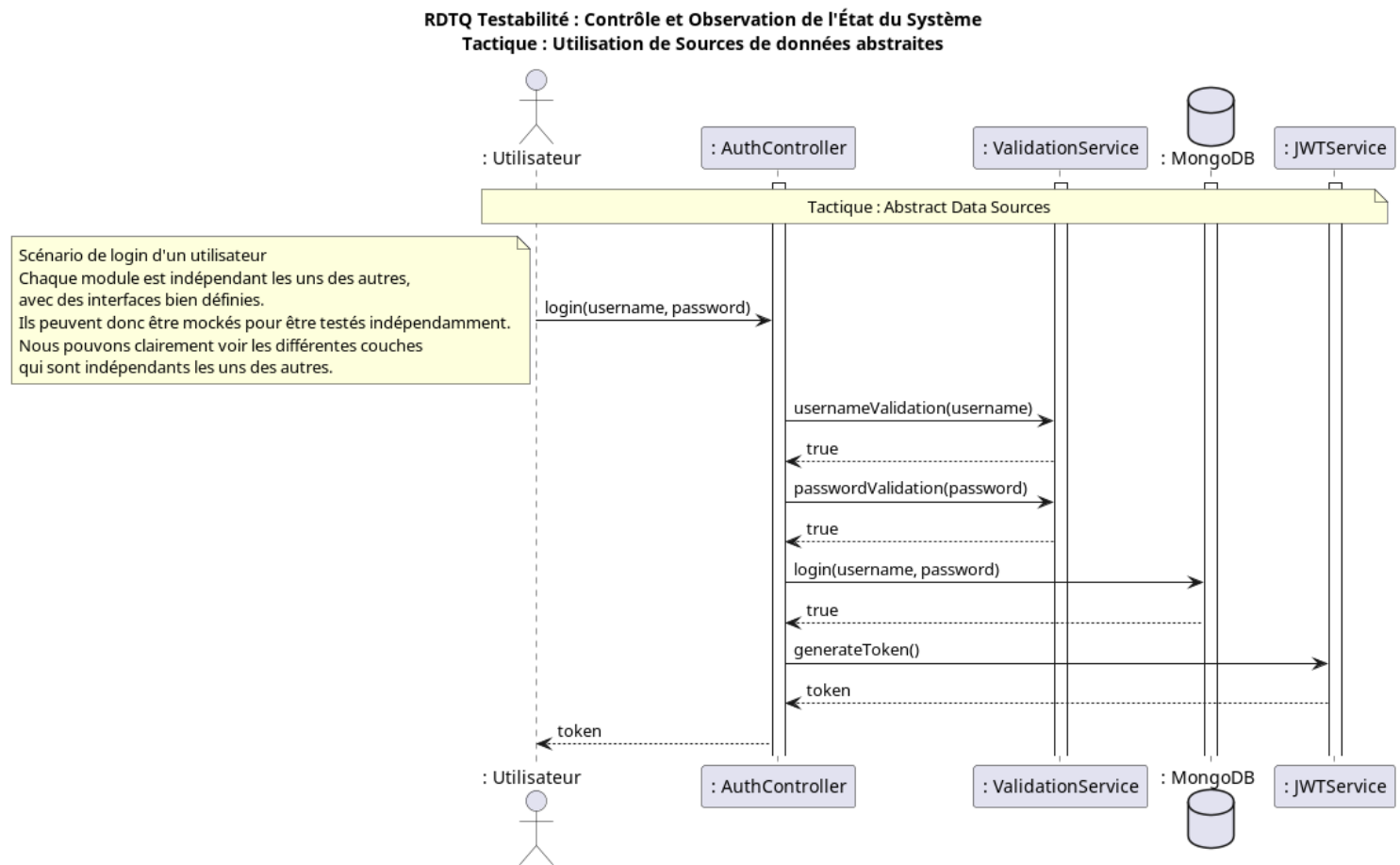
Relation entre les éléments architecturaux et les exigences de sécurité

Identifiant	Éléments	Description de la responsabilité
CU04-S1	Utilisateur	S'enregistrer et s'authentifier dans le système
CU04-S2	Authentication Controller	Gestion de la sécurité dans l'authentification

RDAQ-Testabilité

RDTQ-Contrôle et observe l'état du système

Tactique : Utilisation de Sources de données abstraites

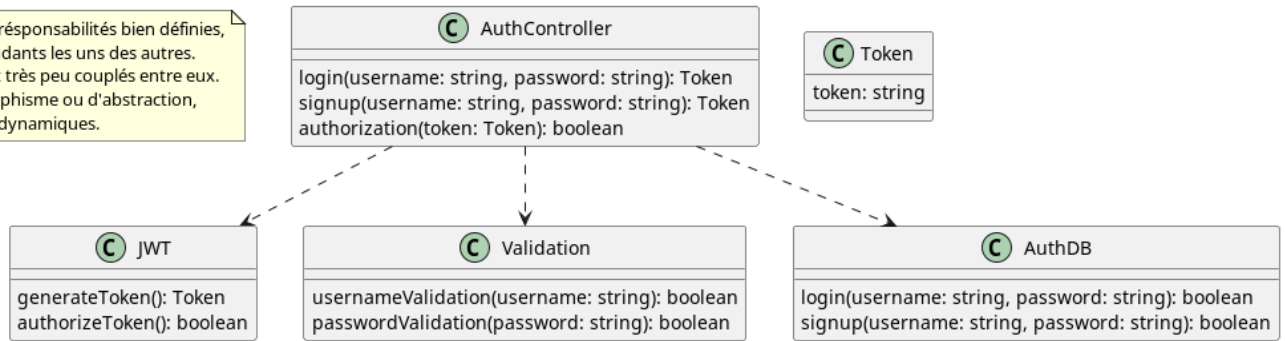


RDTQ-limiter la complexité

Tactique: Limiter la Complexité Structurale

RDTQ Testabilité : Limiter la Complexité du Système
Tactique: Limiter la Complexité Structurale

Chaque module a des responsabilités bien définies, et sont très peu dépendants les uns des autres. Ils sont très cohésifs et très peu couplés entre eux. Il n'y a pas de polymorphisme ou d'abstraction, seulement des appels dynamiques.



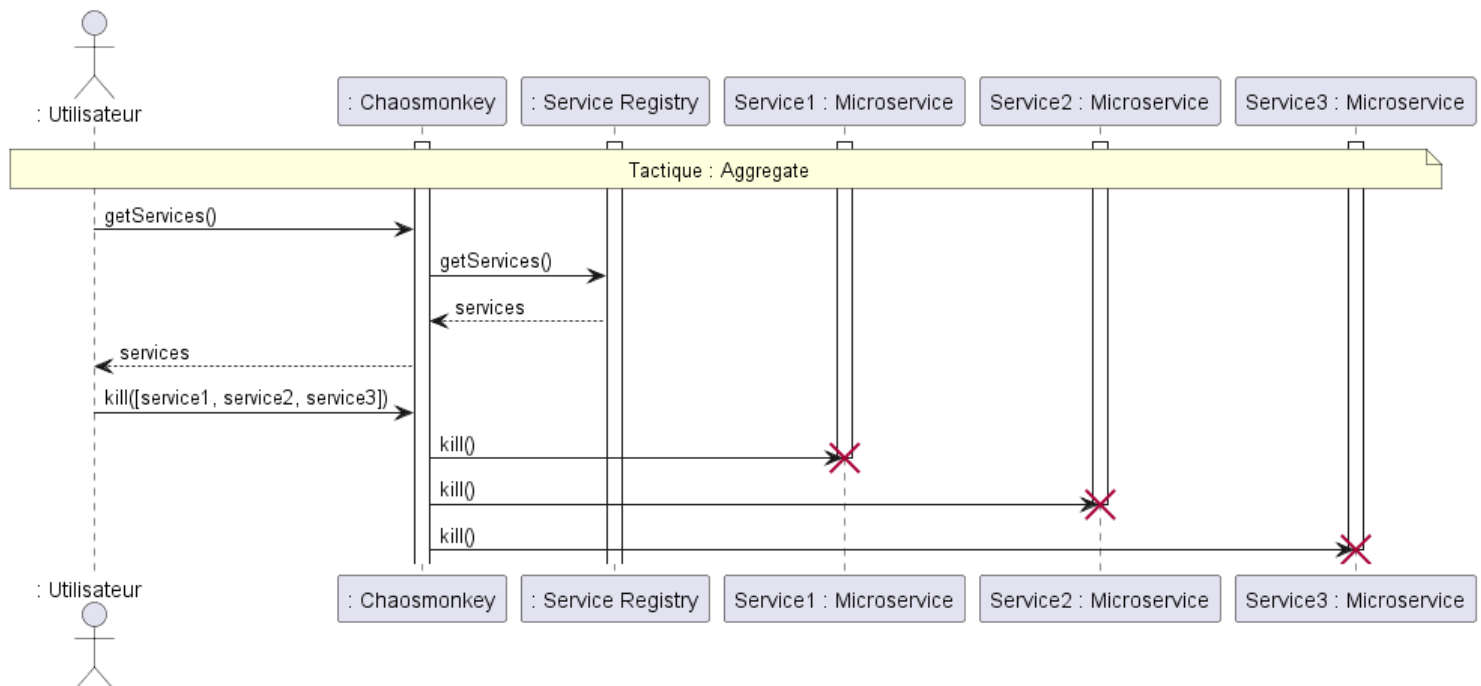
Relation entre les éléments architecturaux et les exigences de testabilité

Identifiant	Éléments	Description de la responsabilité
CU04-T1	Authentification	Microservice d'authentification et d'autorisation du système

RDAQ-Usabilité

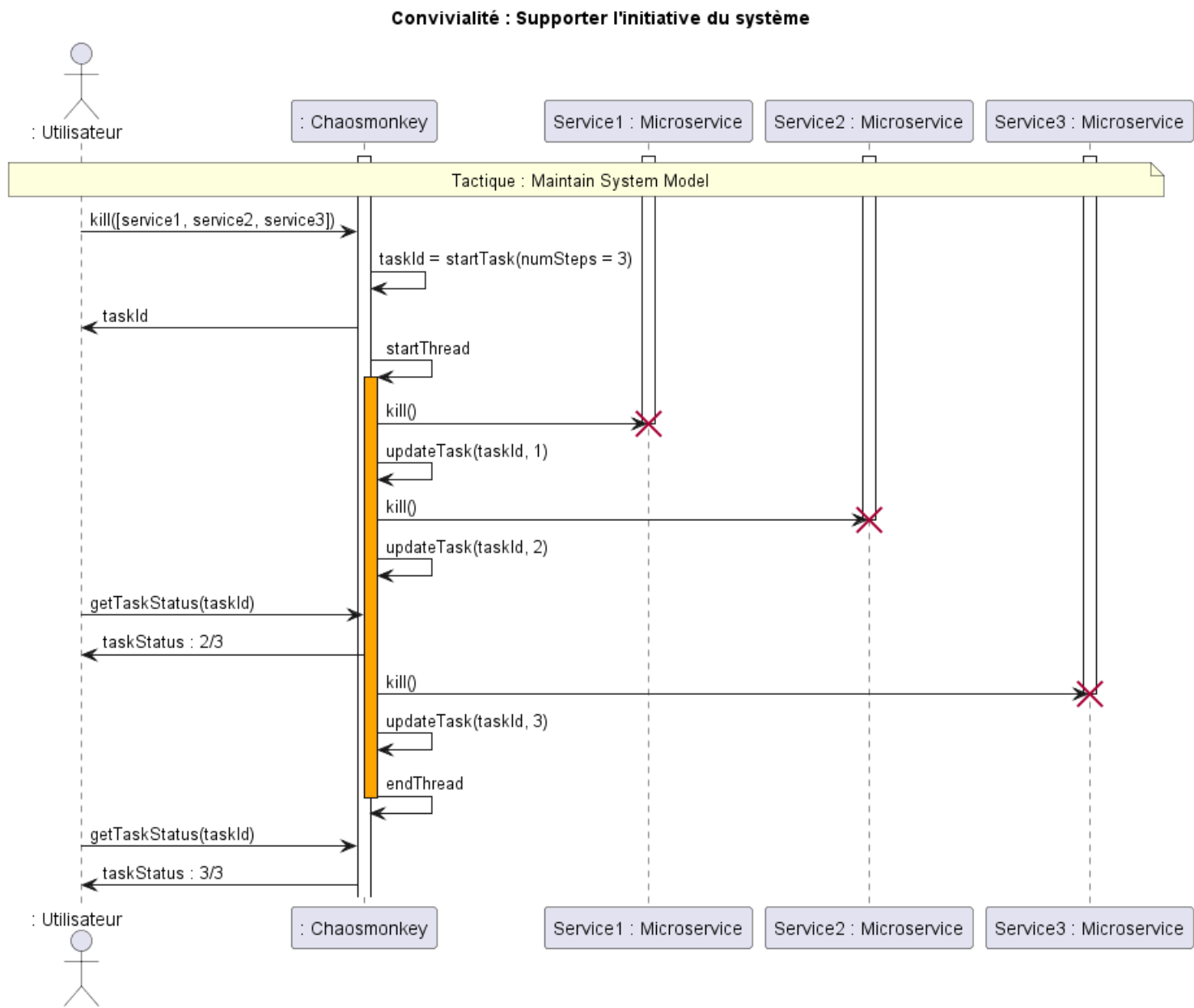
RDTQ-Supporter l'initiative de l'utilisateur

Convivialité : Supporter l'initiative de l'utilisateur



Afin de faciliter l'utilisation de l'application, on permet à l'utilisateur de combiner plusieurs requêtes.

RDTQ-Supporter l'initiative du système



On garde l'état d'une tâche afin de pouvoir informer l'utilisateur sur sa progression.

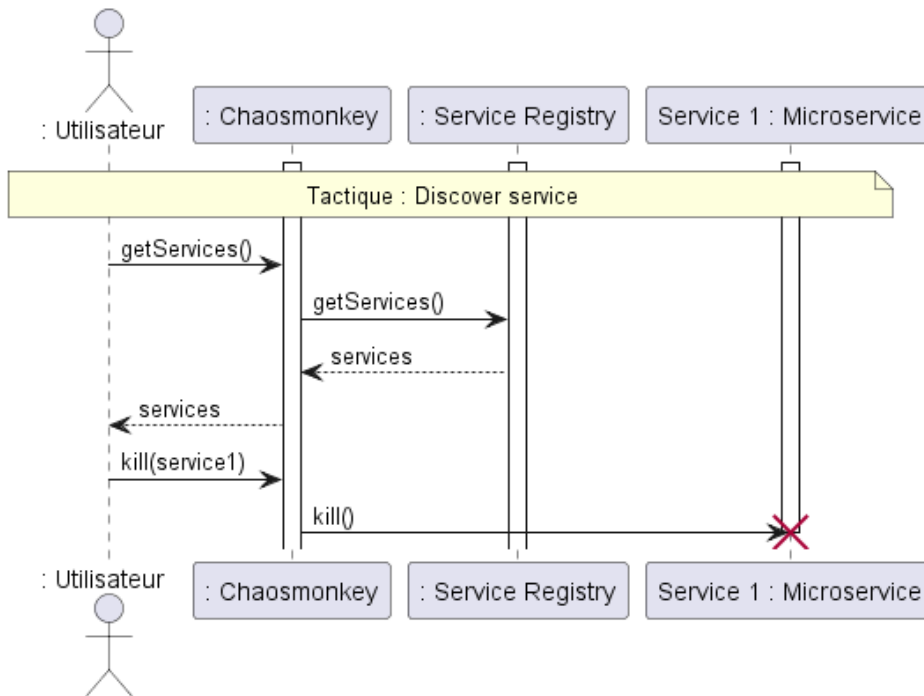
Relation entre les éléments architecturaux et les exigences d'usabilité

Identifiant	Éléments	Description de la responsabilité
CU02-U1	Chaosmonkey	Doit permettre à l'utilisateur d'aggréger ses requêtes et de voir l'état d'une requête
CU02-U2	Service registry	Doit fournir la liste des services
CU02-U3	Utilisateur	Est celui à qui on veut fournir une bonne expérience

RDAQ-Interopérabilité

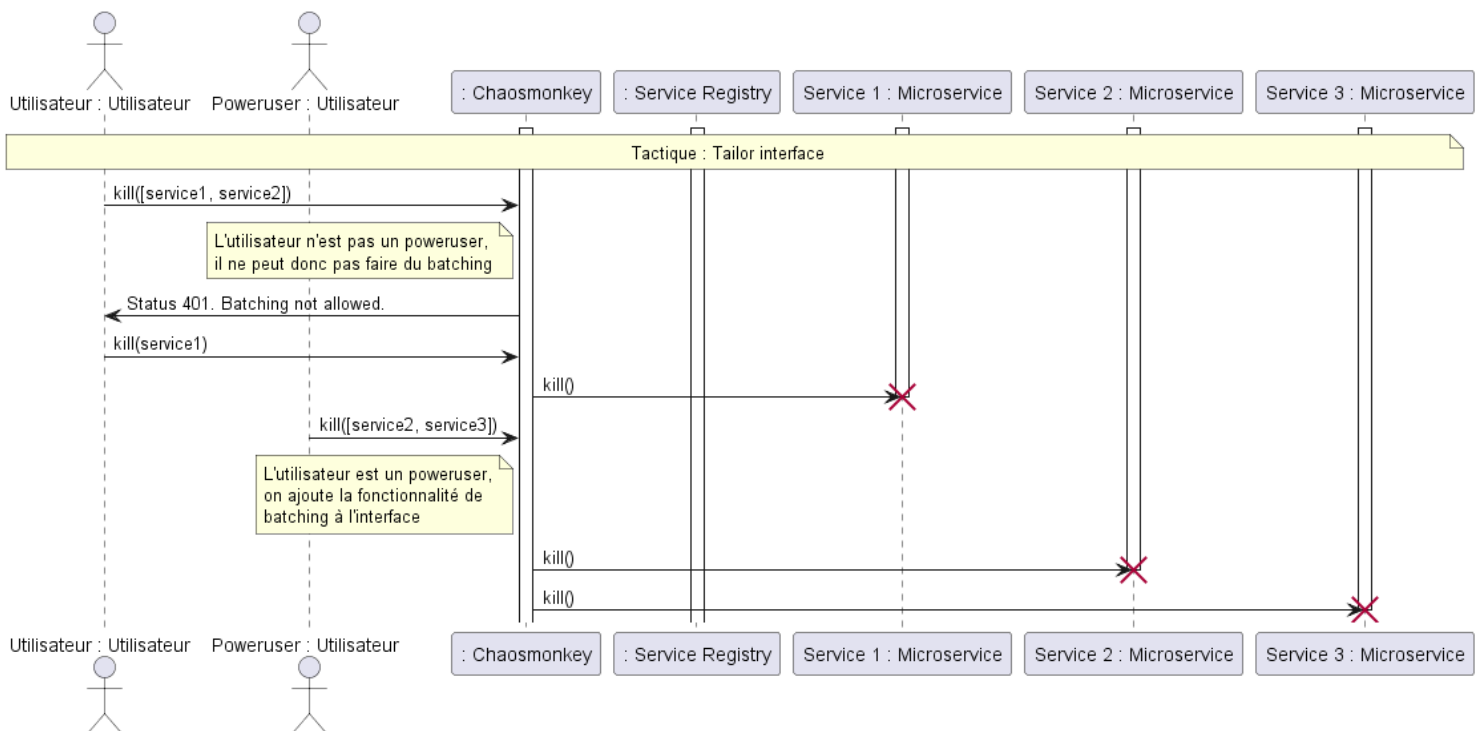
RDTQ-Localiser

RDTQ Interopérabilité : Localiser



RDTQ-Gérer les interfaces

RDTQ Interopérabilité : Gérer les interfaces



Afin de permettre à différents utilisateurs d'avoir accès à différentes fonctionnalités sur les mêmes routes, on ajoute la fonctionnalité d'aggrégation aux "Powerusers".

Relation entre les éléments architecturaux et les exigences d'interopérabilité

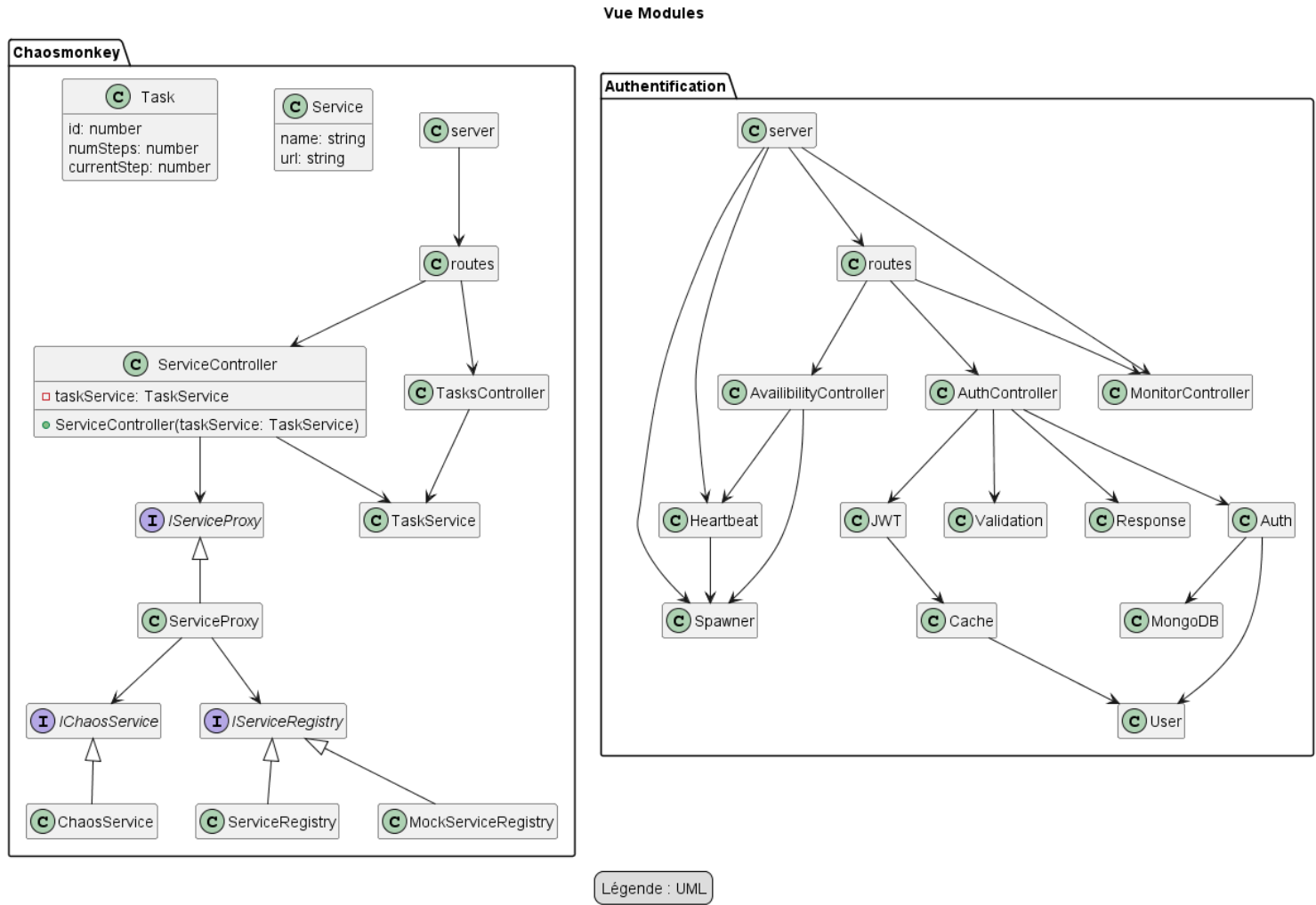
Identifiant	Éléments	Description de la responsabilité
CU02-I1	Utilisateur	N'est pas autorisé à agréger ses requêtes
CU02-I2	Poweruser	Est autorisé à agréger ses requêtes
CU02-I3	Chaosmonkey	Gère ses interfaces
CU02-I4	Service Registry	Fournit la liste des services

Vues architecturales

Vues architecturales de type Module

Vue #1

Présentation primaire

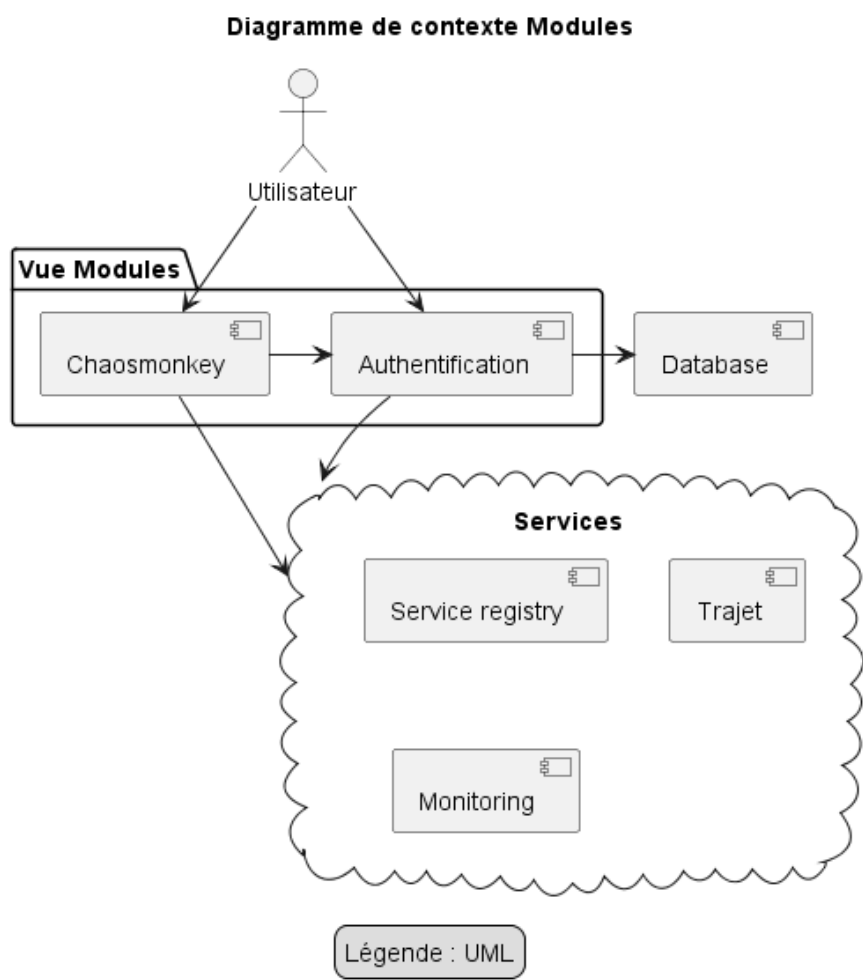


Catalogue d'éléments

Élément	Description	lien vers document d'interfaces
Authentification.AuthController	Contrôleur responsable de générer et des valider les jetons d'authentification	Document d'interface Authentification
Authentification.AvailabilityController	Contrôleur responsable de gérer les heartbeats entre les différentes copies	Document d'interface Authentification

Élément	Description	lien vers document d'interfaces
Authentification.MonitorController	Contrôleur responsable d'obtenir l'état des microservices (incluant celui-ci) et de simuler les pannes en répondant au Chaosmonkey	Document d'interface Authentification
Chaosmonkey.ServiceController	Contrôleur responsable d'obtenir la liste des microservices et d'envoyer les requêtes aux autres microservices afin de simuler les pannes	Document d'interface Chaosmonkey
Chaosmonkey.TasksController	Contrôleur responsable d'obtenir l'état des tâches prenant beaucoup de temps	Document d'interface Chaosmonkey

Diagramme de contexte



Guide de variabilité

Puisque ce diagramme ne présente que les modules présents au niveau de l'implémentation, ils n'ont pas nécessairement de présence au moment de l'exécution. À défaut de modifier les fichiers sources de l'application, il n'y a donc pas de variabilité qui s'applique ici.

Raisonnement

Plusieurs contrôleurs ont été développés dans chaque microservice afin de séparer les responsabilités pour les différentes routes de l'application.

Des interfaces internes ont été définies au niveau du Chaosmonkey afin de pouvoir insérer des services "Mock" puisque les autres microservices n'étaient pas disponibles pendant la majorité du temps de développement de l'application.

Vues associées

Tous les diagrammes représentés dans les RDTQ correspondent à des éléments présents dans cette vue.

Vues architecturales de type composant et connecteur

Vue #1

Présentation primaire

VueComposantConnecteurDiagramme

Catalogue d'éléments

Élément	Description	lien vers document d'interfaces
User	Composant qui initie la requête	
Database	Composant qui contient les tokens d'authentification	Document d'interface Authentification
IDatabase	Interface qui utilise le micro service d'authentification pour vérifier la validité d'un token donné par un user	Document d'interface Authentification
Authentification_MS_Originale	Microservice qui authentifie qu'un user a le droit d'accéder à un service donné	Document d'interface Authentification
Authentification_MS_Copie	La copie du micro service originale pour assurer la disponibilité	Document d'interface Authentification
Chaos	Micro service utiliser pour détruire une instance d'un autre microservice	Document d'interface Chaosmonkey

Élément	Description	lien vers document d'interfaces
IserviceDiscovery	Interface à travers laquelle le micro service externe qui intercommunication entre microservices à l'aide d'une source unique de découverte de route est visible	
IserviceTrajet	Interface à travers laquelle le micro service externe qui compare les temps de trajet est visible	
IserviceMonitoring	Interface à travers laquelle le micro service externe qui informe le mainteneur sur le status de vie des autres microservices est visible	

Diagramme de contexte

 VueComposantConnecteurDiagrammeContexte

Guide de variabilité

La vue composant connecteur, présente tous les éléments obligatoires pour faire fonctionner le système. Donc nous n'avons pas de variabilité.

Raisonnement

En ce qui concerne la vue composant et connecteur, nous avons pris pour hypothèse qu'on a une seule copie du micro service d'authentification qui reçoit les requêtes provenant des utilisateurs et du micro service chaos. Dépendamment de la requête, il va soit interroger la base de données pour vérifier leur authenticité soit transférer la requête vers interface public qui lui connaît tous les autres micro services externes associés.

Vues associées

[Document d'interface Authentification](#)

Vues architecturales de type allocation

Vue #1 - Déploiement pour authentification

Présentation primaire

 VueAllocationDiagramme

Catalogue d'éléments

Élément	Description	lien vers document d'interfaces
PC utilisateur (externe)	Élément physique: ordinateur d'un utilisateur qui se connecte au service	Vue architecturale de contexte
PC utilisateur (interne)	Élément physique: ordinateur d'un utilisateur qui utilise le service sur sa machine	Vue architecturale de contexte
Serveur	Élément physique: machine qui permet d'exécuter le service	Vue architecturale de contexte
Application d'authentification	Élément logiciel: service d'authentification	Vue architecturale de contexte
Base de données	Élément physique: élément qui garde en mémoire les informations des utilisateurs	Vue architecturale de contexte

Diagramme de contexte



Guide de variabilité

Dans cette vue d'allocation, il n'y a pas de variabilité évidente. Tous les éléments doivent obligatoirement être présents pour le bon fonctionnement du système, ils ne peuvent pas être changés et l'implémentation non plus.

Raisonnement

En ce qui a trait aux décisions par rapport à la vue allocation, ici, nous avons fait l'hypothèse qu'un seul serveur est suffisant pour supporter les deux microservices et tous les utilisateurs qui voudraient y accéder en même temps. Nous avons aussi assumé qu'étant donnée l'étendue très petite de ce service, une seule copie de la base de donnée serait utile.

Vues associées

[Vue architecturale de contexte](#)

Conclusion

Dans ce projet, nous avons eu à développer plusieurs fonctionnalités dans le contexte d'une architecture composée de microservices ayant pour but principal de fournir un service de calcul de temps de trajet.

Nous avons échangé de la documentation avec d'autres équipes afin d'assurer la compatibilité des différents services, et avons implémenté et intégré plusieurs tactiques architecturales afin d'assurer le respect des différents attributs de qualité.

Nous avons ainsi obtenu deux microservices stables, simples à comprendre, et efficaces.

Ce projet nous a permis de développer une application relativement complexe dans un contexte s'apparentant au développement logiciel dans une grosse entreprise, où différents départements travaillent sur un même projet en échangeant leur documentation publique.

Documentation des interfaces

Élément	Description	lien vers document d'interfaces
Chaos Monkey	Ce microservice a été créé pour perturber les autres microservices, en simulant une panne. Il est utilisé pour les tuer, forçant la bonne implémentation du redémarrage ainsi que de s'assurer de la disponibilité du service.	Document d'interface Chaosmonkey
Authentification	Ce service d'authentification et d'autorisation sert à vérifier que l'utilisation du système par un utilisateur est sincère est justifié. L'utilisateur concerné doit alors s'authentifier dans le système avant de pouvoir effectuer une action, qui devra être autorisée par ce microservice. Les informations essentielles seront stockées dans une base de données afin d'assurer la persistance et la continuité du service.	Document d'interface Authentification