Documentation de l'architecture du laboratoire de LOG430

        - CU05-D1 **Disponibilité**
        - CU05-M1 **Modifiabilité**
        - CU05-P1 **Performance**
        - CU05-S1 **Sécurité**
        - CU05-T1 **Testabilité**
        - CU05-U1 **Usabilité**
        - CU05-I1 **Interopérabilité**
    - **CU06** - Veux créer trajet STM
        - CU06-D1 **Disponibilité**
        - CU06-M1 **Modifiabilité**
        - CU06-P1 **Performance**
        - CU06-S1 **Sécurité**
        - CU06-T1 **Testabilité**
        - CU06-U1 **Usabilité**
        - CU06-I1 **Interopérabilité**
    - **CU07** - Veux calculer temps trajet automobile
        - CU07-D1 **Disponibilité**
        - CU07-M1 **Modifiabilité**
        - CU07-P1 **Performance**
        - CU07-S1 **Sécurité**
        - CU07-T1 **Testabilité**
        - CU07-U1 **Usabilité**
        - CU07-I1 **Interopérabilité**
    - **CU08** - Veux calculer temps trajet STM
        - CU08-D1 **Disponibilité**
        - CU08-M1 **Modifiabilité**
        - CU08-P1 **Performance**
        - CU08-S1 **Sécurité**
        - CU08-T1 **Testabilité**
        - CU08-U1 **Usabilité**
        - CU08-I1 **Interopérabilité**
    - **CU09** - Veux avoir la météo
        - CU09-D1 **Disponibilité**
        - CU09-M1 **Modifiabilité**
        - CU09-P1 **Performance**
        - CU09-S1 **Sécurité**
        - CU09-T1 **Testabilité**
        - CU09-U1 **Usabilité**
        - CU09-I1 **Interopérabilité**
    - **CU10** - Veux pouvoir centraliser les services du système
        - CU10-D1 **Disponibilité**
        - CU10-M1 **Modifiabilité**
        - CU10-P1 **Performance**
        - CU10-S1 **Sécurité**
        - CU10-T1 **Testabilité**
        - CU10-U1 **Usabilité**

        - CU10-I1 **Interopérabilité**
- Vue architecturale de contexte
    - Présentation primaire
    - Catalogue d'éléments
    - Diagramme de contexte Pas nécessaire puisque c'est déja un vue de contexte
    - Guide de variabilité
    - Raisonnement
    - Vues associées pas nécessaire puisque c'est la première vue que vous réalisé pour votre système.
- 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-Controle and observe l'état du système
        - ADD-Limiter la complexité
    - ADD-Usabilité
        - ADD-Supporter l'initiative de l'usager
        - 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
    - **RDCU-CU03** - Veux pouvoir vérifier la disponibilité d'un micro-service
    - **RDCU-CU04** - Veux pouvoir sauvegarder un trajet
    - **RDCU-CU05** -
    - **RDCU-CU06** -
    - **RDCU-CU07** -
    - **RDCU-CU08** -
    - **RDCU-CU09** -

Page titre

Introduction

> L'objectif de ce projet est de documenter et développer un système de comparaison de temps de trajet basée sur une architecture de microservices. Ce système permettra de de comparer les temps de trajets entre les autobus et les automobiles. Ce document contient les objectifs d'affaires, les cas d'utilisation, les scénarios de qualité priorisés, des vues architecturales permettant de démonter chacune des tactiques, des vues modules, des vues composants et connecteurs et des vues d'allocation.

Scénario d'objectif d'affaire

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

Notre architecture permet la réalisation de cet objectif d'affaire, car dans notre acrhitecture nous avons mis l'emphase sur la documentation claire ainsi qu'une architecture simple à comprendre afin de faciliter la correction des chargés de laboratoire. En facilitant leurs travail, cela pourra pousser davantages d'étudiants à devenir des chargés de laboratoires.

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

Notre architecture permet la réalisation de cet objectif d'affaire, car il intègre les modules permettant de calculer le temps de trajet par bus et montre la réponse sur une interface graphique. Notre architecture détail davantages les modules permettant d'assurer la disponibilité du module de calcul de trajets ainsi que celui permettant de sauvegarder des trajets spécifiques selon un utilisateur.

Cas d'utilisations

CU01 - Veux comparer les temps de trajet.

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 ce cas d'utilisation est terminé et l'équipe demande au chargé de laboratoire de corriger celle-ci.
- L'intégration est complété et l'équipe demande au chargé de laboratoire de corriger celle-ci
- L'implémentation est complété est l'équipe demande au chargé de laboratoire de corriger celle-ci.

Scénario

1. Le (chargé de laboratoire) CL sélectionne une intersection de départ et une intersection d'arrivée, ainsi que le taux de rafraichissement de la prise de mesure.
2. Le CL sélectionne le service externe qu'il veut utiliser pour faire la comparaison des temps de trajet avec les donnés temps réel de la STM.
3. Le système affiche un graphique du temps de déplacement et met celui-ci à jour selon le taux de rafraichissement.

Évènement résultant:

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

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

N/a

CU01-M1 Modifiabilité

N/a

CU01-P1 Performance

N/a

CU01-S1 Sécurité

N/a

CU01-T1 Testabilité

N/a

CU01-U1 Usabilité

N/a

CU01-I1 Interopérabilité

N/a

Commentaires:

- Quel sont vos remarques/commentaires par rapport à ce scénario

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 terminé et l'équipe demande au chargé de laboratoire de corriger celle-ci.
- L'intégration est complété et l'équipe demande au chargé de laboratoire de corriger celle-ci
- L'implémentation est complété est 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 perturbé par le mécanisme.
- Le système conserve un log des perturbations
- Le système conserve un log de comment le système a réagit pour résoudre le problème.

Postcondition:

- Les mécanismes de traitement des attributs de qualité détectent le problème et modifie 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

N/a

CU02-M1 Modifiabilité

N/a

CU02-P1 Performance

N/a

CU02-S1 Sécurité

N/a

CU02-T1 Testabilité

N/a

CU02-U1 Usabilité

N/a

CU02-I1 Interopérabilité

N/a

Commentaires:

- Quel sont vos remarques/commentaires par rapport à ce scénario

CU03 - Veux pouvoir vérifier la disponibilité d'un micro-service

Acteurs externe:

- Chargé de laboratoire: Veut pouvoir faire la correction de chaque cas d'utilisation.
- Un micro-service externe au notre.

Précondition:

- Le micro-service du Health Monitor est opérationnel.
- Le micro-service Discovery est opérationnel.

Évènement déclencheur:

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

Scénario

1. Un micro-service appelle notre Health Monitor afin de connaître la disponibilité d'un autre micro-service.
2. Le Health Monitor envoie un ping au micro-service demandé en attendant une réponse.

3. Le Health Monitor renvoi une confirmation indiquant la disponibilité d'un micro-service.

Évènement résultant:

- Le Health Monitor renvoie la disponibilité du micro-service demandé.

Postcondition:

- Le Health Monitor est en attente d'un nouvel appel d'un micro-service.

Cas alternatifs: 2. a Le micro-service ne renvoi pas de réponse au Health Monitor. 2. a1 On répète l'étape 2 cinq fois de suite. 2. a2 Le Health Monitor renvoie un message d'erreur indiquant que le micro-service est indisponible à Discovery.

Attributs de qualité

CU03-D1 Disponibilité

Le service doit être disponible 99.9% du temps.

CU03-M1 Modifiabilité

Il doit être possible d'effectuer une modification mineure sur le module du Health Monitor en dedans de 3 heures.

CU03-P1 Performance

Le module doit pouvoir recevoir une réponse du micro-service en moins de 100ms.

CU03-S1 Sécurité

Le service doit être protéger des attaques DDOS.

CU03-T1 Testabilité

L'on doit pouvoir créer des tests pouvant couvrir 90% du module et ce, en 3 heures.

CU03-U1 Usabilité

L'utilisateur doit pouvoir voir en tout temps que le système est actif et qu'il vérifie l'état des requêtes du système.

CU03-I1 Interopérabilité

L'on doit pouvoir communiquer avec les autres modules de manière à ce que l'information pour contacter les autres services soit centraliser pour tous.

Commentaires:

CU04 - Veux pouvoir sauvegarder un trajet selon l'utilisateur

Acteurs externe:

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

Précondition:

- Le micro-service du Health Monitor est opérationnel.
- Le compte de l'utilisateur à été créé

Évènement déclencheur:

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

Scénario

1. Le (chargé de laboratoire) CL se connecte à son compte en entrant son nom d'utilisateur ainsi que son mot de passe
2. Le système affiche la liste des trajets sauvegardé par l'utilisateur
3. Le CL veut comparer un trajet (voir CU01)
4. Le CL sauvegarde le trajet entrée
5. Le système enregistre le trajet de l'utilisateur et affiche la liste à jour des trajets sauvegardés.

Évènement résultant:

- Le système affiche une liste mise à jour des trajets sauvegardé par le CL

Postcondition:

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

Cas alternatifs:

2. a Le CL envoi un nom d'utilisateur et/ou un mot de passe invalide.
3. a1 Le système affiche que l'entrée utilisateur est erroné.
4. a2 Au bout de cinq essais manqués, le système bloque l'accès du micro-service à l'utilisateur pendant une durée de 5 minutes

Attributs de qualité

CU04-D1 Disponibilité

Le service doit être disponible 99.9% du temps.

CU04-M1 Modifiabilité

Il doit être possible d'effectuer une modification mineure sur le module en moins de 6 heures

CU04-P1 Performance

Le module doit être en mesure de gérer 10 appels clients à la fois sans que le temps de réponse n'excède 800ms.

CU04-S1 Sécurité

Les données des clients ne doivent être accessible que par ceux-ci.

CU04-T1 Testabilité

L'on doit pouvoir créer des tests pouvant couvrir 90% du module et ce, en 6 heures.

CU04-U1 Usabilité

L'utilisateur doit pouvoir supprimer plusieurs trajets en une seule action.

CU04-I1 Interopérabilité

Il doit y avoir des interfaces accessibles pour les autres utilisateurs du micro-service afin d'effectuer des appels sur celui-ci.

Commentaires:

CU05 - Veux créer trajet automobile

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 ce cas d'utilisation est terminé et l'équipe demande au chargé de laboratoire de corriger celle-ci.
- L'intégration est complété et l'équipe demande au chargé de laboratoire de corriger celle-ci
- L'implémentation est complété est l'équipe demande au chargé de laboratoire de corriger celle-ci.

Scénario

1. Le (chargé de laboratoire) CL sélectionne une intersection de départ et une intersection d'arrivée.
2. Le système affiche que le trajet a bien été créé et l'ajoute dans la liste.

Évènement résultant:

- Le système affiche une liste des trajets automobiles qui sont crées.

Postcondition:

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

Cas alternatifs:

1. a Le CL sélectionne une intersection de départ, des intersections passantes et une intersection d'arrivée.

Attributs de qualité

CU05-D1 Disponibilité

N/a

CU05-M1 Modifiabilité

N/a

CU05-P1 Performance

N/a

CU05-S1 Sécurité

N/a

CU05-T1 Testabilité

N/a

CU05-U1 Usabilité

N/a

CU05-I1 Interopérabilité

N/a

Commentaires:

CU06 - Veux créer trajet STM

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 ce cas d'utilisation est terminé et l'équipe demande au chargé de laboratoire de corriger celle-ci.
- L'intégration est complété et l'équipe demande au chargé de laboratoire de corriger celle-ci
- L'implémentation est complété est l'équipe demande au chargé de laboratoire de corriger celle-ci.

Scénario

1. Le (chargé de laboratoire) CL sélectionne une intersection de départ et une intersection d'arrivée.
2. Le système affiche que le trajet a bien été créé et l'ajoute dans la liste.

Évènement résultant:

- Le système affiche une liste des trajets automobiles qui sont crées.

Postcondition:

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

Cas alternatifs:

1. a Le CL sélectionne une intersection de départ, des intersections passantes et une intersection d'arrivée.

Attributs de qualité

CU06-D1 Disponibilité

N/a

CU06-M1 Modifiabilité

N/a

CU06-P1 Performance

N/a

CU06-S1 Sécurité

N/a

CU06-T1 Testabilité

N/a

CU06-U1 Usabilité

N/a

CU06-I1 Interopérabilité

N/a

Commentaires:

CU07 - Veux calculer temps trajet automobile

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 ce cas d'utilisation est terminé et l'équipe demande au chargé de laboratoire de corriger celle-ci.
- L'intégration est complété et l'équipe demande au chargé de laboratoire de corriger celle-ci
- L'implémentation est complété est l'équipe demande au chargé de laboratoire de corriger celle-ci.

Scénario

1. Le (chargé de laboratoire) CL sélectionne un trajet automobile.
2. Le système affiche un graphique du temps de déplacement.

Évènement résultant:

- Le système affiche un graphique du temps de déplacement.

Postcondition:

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

Cas alternatifs:

Attributs de qualité

CU07-D1 Disponibilité

N/a

CU07-M1 Modifiabilité

N/a

CU07-P1 Performance

N/a

CU07-S1 Sécurité

N/a

CU07-T1 Testabilité

N/a

CU07-U1 Usabilité

N/a

CU07-I1 Interopérabilité

N/a

Commentaires:

CU08 - Veux calculer temps trajet STM

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 ce cas d'utilisation est terminé et l'équipe demande au chargé de laboratoire de corriger celle-ci.
- L'intégration est complété et l'équipe demande au chargé de laboratoire de corriger celle-ci
- L'implémentation est complété est l'équipe demande au chargé de laboratoire de corriger celle-ci.

Scénario

1. Le (chargé de laboratoire) CL sélectionne un trajet STM.
2. Le système affiche un graphique du temps de déplacement.

Évènement résultant:

- Le système affiche un graphique du temps de déplacement.

Postcondition:

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

Cas alternatifs:

Attributs de qualité

CU08-D1 Disponibilité

N/a

CU08-M1 Modifiabilité

N/a

CU08-P1 Performance

N/a

CU08-S1 Sécurité

N/a

CU08-T1 Testabilité

N/a

CU08-U1 Usabilité

N/a

CU08-I1 Interopérabilité

N/a

Commentaires:

CU09 - Veux avoir la météo

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 ce cas d'utilisation est terminé et l'équipe demande au chargé de laboratoire de corriger celle-ci.
- L'intégration est complété et l'équipe demande au chargé de laboratoire de corriger celle-ci
- L'implémentation est complété est l'équipe demande au chargé de laboratoire de corriger celle-ci.

Scénario

1. Le CL sélectionne une date et une heure de départ.
2. Le système affiche dans une interface graphique la température, l'humidité et si le ciel est dégagé.

Évènement résultant:

- Le système affiche la température de la date désirée sur l'île de Montréal.

Postcondition:

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

Cas alternatifs:

2. a La date ou l'heure de départ est invalide, l'interface affiche une erreur.

Attributs de qualité

CU09-D1 Disponibilité

N/a

CU09-M1 Modifiabilité

N/a

CU09-P1 Performance

N/a

CU09-S1 Sécurité

N/a

CU09-T1 Testabilité

N/a

CU09-U1 Usabilité

N/a

CU09-I1 Interopérabilité

N/a

Commentaires:

CU10 - Veux pouvoir centraliser les services du système

Acteurs externe:

- **Microservice:** Veut pouvoir s'enregistrer au service de discovery et avoir les informations des autres services.
- **App core** Veut avoir les informations des autres services.

Précondition:

- Tous les microservices sont opérationnels

Évènement déclencheur:

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

Scénario

1. Le microservice s'enregistre auprès du service de discovery
2. Il reçoit une réponse comme quoi il s'est bien enregistré

Évènement résultant:

- Le système est maintenant enregistré auprès du discovery service.

Postcondition:

- Le système est en attente.

Cas alternatifs:

1. a L'application core veut connaître les informations d'un autres services.
2. a L'application core connaît maintenant l'adresse pour communiquer à un autre microservice et peut faire traiter de l'information.

Attributs de qualité

CU10-D1 Disponibilité

N/a

CU10-M1 Modifiabilité

N/a

CU10-P1 Performance

N/a

CU10-S1 Sécurité

N/a

CU10-T1 Testabilité

N/a

CU10-U1 Usabilité

N/a

CU10-I1 Interopérabilité

N/a

Commentaires:

Vue architecturale de contexte

Utiliser le gabarit suivant:
https://wiki.sei.cmu.edu/confluence/display/SAD/Template%3AArchitectureViewTemplate

Présentation primaire

Catalogue d'éléments

~~Diagramme de contexte~~ Pas nécessaire puisque c'est déja un vue de contexte

Guide de variabilité

Raisonnement

~~Vues associées~~ pas nécessaire puisque c'est la première vue que vous réalisé pour votre 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	
CU02-D1	
CU03-D1	Pour le service de Monitoring, nous implémenterons une tactique de Removal from service.
CU04-D1	
CU05-D1	
CU06-D1	
CU07-D1	
CU08-D1	
CU09-D1	
CU10-D1	

ADD-détection de faute

Concept de design	Pour	Contre	Valeur	Cout
• Ping / Echo	Simple à implémenter et utilise la télécommunication IP.	N'offre pas la raison pour laquelle le service qu'on veut rejoindre n'est plus disponible.	2	1

Concept de design	Pour	Contre	Valeur	Cout
• Monitor	Délégation de la responsabilité à une entité propre, permet de détecter les pannes de plusieurs façons.	Doit être un service lui-même, s'il tombe en panne, on ne sait plus l'état des autres entités du système.	3	2
• Heartbeat	Assure une performance en communiquant l'état à travers d'autres messages. Plus d'informations dans le message que le Ping/Echo.	Plus difficile à implémenter, chaque service doit s'inscrire au Monitor et savoir comment communiquer avec lui.	2	2
• Timestamp	Permet de savoir comment et quand une panne a lieu puisque chaque stamp est associé à un timestamp.	Peut être fastidieux à implémenter.	1	2
• Sanity Checking	Permet d'avoir des informations précises sur la cause de la panne.	Nécessite une bonne connaissance du service et du flow de ses opérations.	2	2
• Condition Monitoring	Prévient les comportements qui peuvent causer des pannes.	Peut introduire de nouvelles erreurs si le système est trop complexe.	3	3
• Voting	Prévient les pannes en utilisant plusieurs fois le même système en parallèle.	Utilise plus de ressources et on doit implémenter une certaine logique pour savoir quelle information est la bonne lorsque les sorties ne concordent pas.	3	3
• Exception Detection	Lève une exception lorsqu'un comportement fautif se produit, permet de garder en ligne le service.	Difficile de prévenir tous les comportements fautifs.	2	2

Concept de design	Pour	Contre	Valeur	Cout
• Self-test	Teste les procédures d'un sous-système, facile de corriger à l'avance les problèmes qui peuvent survenir plus tard.	Long à implémenter.	2	2

Quelle tactique avez vous choisi et pourquoi?

Nous avons choisis la tactique Monitor pour notre micro-service de gestion de trajets selon un utilisateur. Nous avons implémenté le micro-service de monitoring alors il sera plus simple d'implémenter cette tactique.

ADD-Préparation et réparation

Concept de design	Pour	Contre	Valeur	Cout
• Redondance active	Les noeuds redondants processent en parallèle les inputs ce qui fait que lorsqu'un défaut survient sur le module originale la transition se fait de façon instantanée.	Besoin de plus de ressources pour processer les inputs en double, Il faut aussi implémenter la façon que les noeuds redondants procèssent les informations aussi.	3	3
• Redondance passive	Seulement certains noeuds sont redondants et reçoivent des mises à jour périodiques. La redondance passive est moins complexe et celle active et coûte moins cher.	Assure moins la disponibilité que la redondance active.	2	2
• Spare	Les noeuds redondants sont activés seulement en cas de défauts, coûte encore moins cher.	Mauvaise performance de récupération.	1	2
• Exception handling	Permet de savoir la cause et l'endroit d'une panne. Permet de facilement corriger la source de l'erreur et réessayer.	Plus de codes à écrire.	1	1
• Rollback	Permet de revenir rapidement à un environnement qui fonctionnait.	Nécessite de sauvegarder les états des environnements antérieurs, il faut aussi les stocker à un endroit.	2	2

Concept de design	Pour	Contre	Valeur	Cout
• Software upgrade	Les correctifs pour les clases et les fonctions permettent de fixer des bugs alors que le ISSU permet d'offrir des nouvelles fonctionnalités.	Très difficile à implémenter.	2	3
• Retry	Assume la faute est transitoire et que le fait de réessayer fonctionnera. Très simple et utile pour les systèmes où les erreurs sont communes.	N'assure pas vraiment une disponibilité.	1	1
• Ignore faulty behavior	Utile pour se protéger des attaques, les comportements à défaut le sont à cause d'un agent extérieur.	N'assure pas une disponibilité pour les problèmes internes.	1	1
• Degradation	Permet de garder certaines fonctionnalités du système en cas de défaut.	Certaines fonctionnalités ne sont plus disponibles, comment les réintroduire?	2	3
• Reconfiguration	Permet de garder certaines fonctionnalités du système en cas de défaut.	Comment déterminer les responsabilités à reconfigurer et réassigner? Complexe.	2	2

Quelle tactique avez vous choisi et pourquoi?

Nous avons choisis la tactique Exception handling pour notre micro-service de gestion de trajets selon un utilisateur. Notre service aura plusieurs fonctionnalités alors il sera important de savoir l'endroit de la source d'erreur.

ADD-Réintroduction

Concept de design	Pour	Contre	Valeur	Cout
• Shadow	Permet de réintroduire le service en retrait pour repopuler de façon incrémentale et de monitorer.	Il faut implémenter la logique de réintroduction incrémentale.	2	2
• State Resynchronization	Permet de réintroduire le service en utilisant la redondance active. Assure la véracité des données traitées.	Complexe à implémenter.	3	3

Concept de design	Pour	Contre	Valeur	Cout
• Escalating restart	Permet de réintroduire le service granulairement, par phases. Minimise le niveau de services affectés.	Complexe à implémenter.	3	3
• Non stop forwarding	Permet de continuer le transfert de paquets sur les routes connues, en attendant la réparation des tables de routage.	Plus au niveau réseautique.	3	3

Quelle tactique avez vous choisi et pourquoi?

Nous avons choisis la tactique Escalating restart pour notre micro-service de gestion de trajets selon un utilisateur. Ce sera utile pour notre micro-service de gestion de trajets selon un utilisateur, parce que notre service peut être divisé en plusieurs morceaux et ceux-ci peuvent être réintroduits progressivement.

ADD-Prévention des fautes

Concept de design	Pour	Contre	Valeur	Cout
• Removal from service	Simple à implémenter, il faut juste bien détecter les évènements qui pourraient causer un défaut parce que sinon on empêche l'utilisation de notre service à nos utilisateurs.	Même si pour cette tactique l'objectif est de prévenir les défauts en mettant volontairement des composants de notre système out-of-service brièvement nous empêchons quand même pendant quelques secondes l'utilisateur d'utiliser le service.	2	1

Concept de design	Pour	Contre	Valeur	Cout
• Transactions	Le concept ACID est une bonne règle à suivre pour avoir un service dont les données sont toujours cohérentes et que celles-ci ne causent pas de problèmes de disponibilité.	S'assurer d'avoir des échanges ACID entre nos systèmes peut être complexe à implémenter par exemple, avoir un système de rollback lorsqu'un échange cause une erreur à la fin de son éxécution est long à implémenter.	2	2
• Predictive model	Permet de réagir correctement à plusieurs éventualités qui peuvent causer un défaut.	Difficile de définir et construire un modèle de prédiction, sur quelles variables doit on nous baser?	2	3
• Exception prevention	Permet de prévenir les exceptions de systèmes. Évite les fuites de mémoire.	Difficile à implémenter.	2	2
• Increase competence set	Donne aux programmes les outils pour gérer les états en dehors de son ensemble de compétences. Va lancer exception pour avertir les autres composants du système sans tout faire planter.	Difficile de prévoir tous les types d'exceptions.	1	2

Quelle tactique avez vous choisi et pourquoi?

Nous avons choisis la tactique Removal from service puisqu'elle est quand même simple à implémenter. Notre service étant uniquement utilisable par des requêtes GET, il sera facile de contrôler les appels et de réagir en conséquence.

ADD-Modifiabilité

Identifiant	Description
CU01-M1	
CU02-M1	
CU03-M1	Pour le monitoring, nous avons choisi la tactique du defer binding.
CU04-M1	
CU05-M1	

Identifiant	Description
CU06-M1	
CU07-M1	
CU08-M1	
CU09-M1	
CU10-M1	

ADD-Réduire la taille des modules

Concept de design	Pour	Contre	Valeur	Cout
• Split module	Séparer le code en modules permet une meilleure flexibilité lors d'ajout de fonctionnalités.	Développement un peu plus long.	2	1

Quelle tactique avez vous choisi et pourquoi?

Nous avons choisis la tactique Split module pour notre micro-service de gestion de trajets selon un utilisateur. Notre micro-service aura plusieurs responsabilités et il sera primordiale de bien séparer les modules pour permettre une bonne extensibilité des fonctionnalités.

ADD-Augmenter la cohésion

Concept de design	Pour	Contre	Valeur	Cout
• Increase semantic coherence	Permet une meilleure séparation des responsabilités, rend le code plus facile à lire et comprendre.	Peut augmenter le couplage.	3	2

Quelle tactique avez vous choisi et pourquoi?

Nous avons choisis la tactique Increase semantic coherence pour notre micro-service de gestion de trajets selon un utilisateur. Notre micro-service aura plusieurs responsabilités et celles-ci devront être bien séparées pour assurer une bonne cohésion.

ADD-Réduire le couplage

Concept de design	Pour	Contre	Valeur	Cout
• Encapsulation	Permet la modification de modules sans propager les changements dans d'autres modules.	Limite la flexibilités des interactions venant de l'extérieur du module.	2	1

Concept de design	Pour	Contre	Valeur	Cout
• Intermédiaire	Brise une mauvaise dépendance entre deux modules.	Implémentation de plus pour faire la médiation.	3	2
• Restriction des dépendances	Empêche l'utilisation de modules hors de la portée du module concerné.	Nécessite une bonne structure des modules pour ne pas empêcher une bonne dépendance.	2	2
• Refactorisation	Réduit la répétition de code.	Peut rendre le code difficile à lire.	2	2
• Abstraction des services communs	Réduit la répétition de code. Nécessite la modification à un endroit à la place de plusieurs.	Peut rendre le code difficile à lire.	2	2

Quelle tactique avez vous choisi et pourquoi?

Nous avons choisis la tactique intermédiaire pour notre micro-service de gestion de trajets selon un utilisateur. Notre micro-service sera utilisé par d'autres applications alors notre système s'enregistrera au près d'un service de discovery.

ADD-Defer binding

Concept de design	Pour	Contre	Valeur	Cout
• Publish-subscribe	Permet d'avoir l'information relié à un évènement sans connaître la source du dit évènements. Réduit le couplage et permet une meilleure extensibilité.	La file de messages peut être encombré et causer des ralentissements.	3	2

Quelle tactique avez vous choisi et pourquoi?

Nous utiliserons la patron architecturale Publish-subscribe pour aller chercher les informations des nouveaux micro-services qui s'enregistrent auprès du service de Discovery.

ADD-Performance

Identifiant	Description
CU01-P1	
CU02-P1	
CU03-P1	Pour le monitoring, nous avons choisi la tactique Maintain multiple copies of computations

Identifiant	Description
CU04-P1	
CU05-P1	
CU06-P1	
CU07-P1	
CU08-P1	
CU09-P1	
CU10-P1	

ADD-Contrôler la demande en ressources

Concept de design	Pour	Contre	Valeur	Cout
• Manage sampling rate	Utile pour traiter un très grand nombre de données, lorsque ceux-ci peuvent être ignorées pour avoir une meilleure performance.	Mauvais si la fidélité des données est un aspect très important.	2	2
• Limit event response	Utile pour traiter beaucoup de données sur un grand lapse de temps, les données sont dans une file d'attente et seront traitées lorsque ce sera leur tour.	Peut prendre du temps à traiter toutes les données, certaines requêtes importantes peuvent être bloquées longtemps.	1	2
• Prioritize events	Utile pour traiter beaucoup de données et garder une certaine performance. Priorise les requêtes jugées plus importantes de la file en premier.	Il faut définir sous quelles conditions quels évènements sont plus importants.	2	1
• Reduce overhead	Meilleure performance parce qu'on enlève l'utilisation d'intermédiaires.	Perte de cohésion, traitement centralisé, moins extensible.	2	2

Concept de design	Pour	Contre	Valeur	Cout
• Bound execution times	Utilie pour limiter le temps que peut prendre une requête à être processée. Par le fait même, empêche d'avoir des problèmes de boucles infinies.	Difficile de déterminer quel sera le temps limite. Cela peut aussi cause des résultats moins précis comme par exemple, lorsque énomément de données sont traitées.	2	1
• Increase resource efficiency	Meilleure performance parce que algorithmes mieux concus.	Écrire des algorithmes optimales est difficile.	3	3

Quelle tactique avez vous choisi et pourquoi?

Nous avons choisis la tactique Bound execution times pour limiter le temps qu'une requête peut prendre dans notre système.

Puisque c'est un micro-service qui pingera les autres entités du système, il se peut qu'un système soit out-of-service et par le fait
même ralentisse l'éxécution de la requête à cause de Timeout.

ADD-Gérer les ressources

Concept de design	Pour	Contre	Valeur	Cout
• Increase resources	Meilleure performance parce que plus de ressources.	Coûte cher.	2	3
• Introduce concurrency	Meilleure performance parce que les processus qui traitent la requête fonctionnent en parallèle.	Difficile à implémenter.	3	3
• Maintain multiple copies of computations	Meilleure performance parce que les requêtes peuvent être traitées plusieurs à la fois. Réduit les bloquages sur le même serveur.	Il faut implémenter une logique pour dispatcher les requêtes.	3	3
• Maintain multiple copies of data	Meilleure performance parce que les données sont storées sur plusieurs serveurs, utilisation de caches pour réduire bloquages.	Équipements coûtent cher.	3	3

Concept de design	Pour	Contre	Valeur	Cout
• Bound queue sizes	Limite le nombre de requêtes qui peuvent s'accumuler dans la file alors assure une performance.	Il faut définir ce qui arrive lorsque la file déborde, on fait quoi avec les nouvelles requêtes?	3	2
• Schedule resources	Réduit la contention sur les ressources, le réseau, les processus.	Il faut définir quoi faire lorsque lorsqu'il y a contention sur une ressource.	2	2

Quelle tactique avez vous choisi et pourquoi?
Nous avons choisis la tactique Maintain multiple copies of computations puisqu'il est quand même facile d'avoir plusieurs
container dans portainer.

ADD-Sécurité

Identifiant	Description
CU01-S1	
CU02-S1	
CU03-S1	Pour le monitoring, on a choisi la tactique detect service denial.
CU04-S1	
CU05-S1	
CU06-S1	
CU07-S1	
CU08-S1	
CU09-S1	
CU10-S1	

ADD-Détecter les attaques

Concept de design	Pour	Contre	Valeur	Cout
• Detect intrusion	Permet de se protéger des types d'intrusion commun, puisqu'on utilise une base données contenant les signatures du traffic réseau.	Très difficile à implémenter, doit connaître a priori les types d'intrus.	1	2

Concept de design	Pour	Contre	Valeur	Cout
• Detect service denial	Protège d'une attaque DDoS qui est sûrement l'une des types d'attaque les plus facile à exercer. Il suffit de surcharger un serveur d'un nombre de requêtes impossible à traiter.	Ne protège pas des attaques plus sournoises qui peuvent soutirer de l'information sensible.	2	2
• Verify message integrity	S'assure que le message n'a pas été modifié en cours de route par un parti tiers qui s'en sert pour infiltrer le système.	Le système doit maintenir de l'information redondante.	2	2
• Detect message delay	S'assure que le message n'a pas été modifié en cours de route en mesurant le temps que prend le message pour se rendre à destination.	Le message peut être en retard pour plusieurs raisons et non juste des attaques.	1	1

Quelle tactique avez vous choisi et pourquoi?
Puisque notre micro-service de détiendra pas d'informations sur ses utilisateurs et qu'il ne possédera pas d'informations sensibles,
nous avons choisis de prendre la tactique Detect service denial. De cette façon, notre service ne sera pas surcharger et pourra rester online.

ADD-Résister aux attaques

Concept de design	Pour	Contre	Valeur	Cout
• Identify actors	Permet au service de reconnaître la source d'une requête.	À implémenter avec autre tactique. Pas suffisant.	1	1
• Authenticate actors	Assure au système que les requêtes faites par un acteur authentifié sont légitimes.	À implémenter avec autre tactique. Pas suffisant.	1	1

Concept de design	Pour	Contre	Valeur	Cout
• Authorize actors	Donne des privilèges aux acteurs qui sont authentifiés. Assure que ces requêtes reliées aux privilèges sont légitimes.	À implémenter avec autre tactique. Pas suffisant.	2	1
• Limit access	Bloque certains accès critiques aux systèmes, comme la mémoire et les connexions réseaux.	Certaines accès sont bloqués à des acteurs légitimes.	2	2
• Limit exposure	Réduit le nombre d'accès possibles et par le fait même, la probabilités d'une attaque.	Ne protège pas activement le système.	2	2
• Encrypt data	Protège la communication d'informations sensibles sur les liens publiques.	Il faut implémenter le système d'encryptage (VPN, SSL, clés publiques/privées).	3	2
• Separate entities	Sépare les données sensibles des données non-sensibles pour réduire la possibilité d'attaques venant de ceux qui ont accès aux données non-sensibles.	Séparer physiquement nécessite d'autres serveurs/réseaux.	2	3
• Change default settings	Empêche les acteurs malicieux d'avoir accès au système à cause des paramètres par défaut connues publiquement.	À implémenter avec autre tactique. Pas suffisant.	1	2

Quelle tactique avez vous choisi et pourquoi?

Nous avons choisis les tactiques authenticate et authorize actors, puisque notre micro-service, qui fera la gestion des trajets selon l'utilisateur, nécessitera une connexion pour savoir quelles trajets sont à qui.

ADD-Réagir aux attaques

Concept de design	Pour	Contre	Valeur	Cout
• Revoke access	Permet de limiter les dégâts d'une attaque.	L'attaque est déjà en court.	1	1

Concept de design	Pour	Contre	Valeur	Cout
• Lock computer	Bloque un ordinateur suspect d'une attaque éventuelle.	Le blocage est temporaire.	2	1
• Inform actors	Permet aux acteurs pouvant aider à une éventuelle attaque de réagir rapidement.	L'attaque est déjà en court.	2	1

Quelle tactique avez vous choisi et pourquoi?

Nous avons choisis la tactique de barrer un ordinateur lorsqu'une activité suspecte est détectée. Ça semble être la seule tactique qui bloque l'attaque, même si c'est temporaire.

ADD-Récupérer d'une attaque

Concept de design	Pour	Contre	Valeur	Cout
• Maintain audit trail	Permet de maintenir un historique des utilisateurs, des actions et des effets sur notre système. Ce qui rend plus facile d'identifier les acteurs et effets d'une attaque.	Comment et où maintenir cette historique?	2	3
• Restore	Qui relève d'une tactique de disponibilité. Permet de restaurer le système.	Peut être complexe à implémenter.	2	2

Quelle tactique avez vous choisi et pourquoi?

Nous avons choisis la tactique restore et plus précisément, le escalating restart. Ce sera utile pour notre micro-service de gestion de trajets selon un utilisateur, parce que notre service peut être divisé en plusieurs morceaux et ceux-ci peuvent être réintroduits progressivement.

ADD-Testabilité

Identifiant	Description
CU01-T1	
CU02-T1	
CU03-T1	Pour le monitoring, nous avons choisi de limiter la complexité de la structure.
CU04-T1	
CU05-T1	
CU06-T1	
CU07-T1	
CU08-T1	

Identifiant	Description
CU09-T1	
CU10-T1	

ADD-Controle and observe l'état du système

Concept de design	Pour	Contre	Valeur	Cout
• Interfaces spécialisées	Permet de tester les méthodes d'un objet pour s'assurer qu'elles font ce qui est désiré. Facile à ajouter ou enlever.	Plus de code à écrire et ne teste pas les inputs de l'environnement d'éxécution.	2	1
• Record/Playback	Permet de recréer facilement l'environnement qui a créé un erreur critique en enregistrant le input et le réutiliser au besoin.	Difficile à implémenter.	2	2
• Localize State Storage	Permet de recréer facilement l'environnement qui a créé un erreur critique en sauvegardant l'état du système.	Difficile à implémenter.	3	2
• Abstraction des sources de données	Permet de tester facilement le système en changeant l'origine des données.	On doit rendre facile le changement de la source des données et avoir une autre base de données.	3	3
• Bac à sable	Permet de faire des tests sans se soucier des conséquences sur le système. Isole une autre instance du service et permet l'expérimentation du système tout en pouvant revenir à l'état d'origine facilement.	Deux instances alors plus de ressources.	2	3
• Executable assertions	Test les valeurs des input et output au fur et à mesure de l'éxécution du code. Rend facile tracer l'évolution de l'état du système.	Peut être fastidieux si le système est complexe et possède beaucoup de lignes de codes.	1	2

Quelle tactique avez vous choisi et pourquoi?

Nous avons choisis la tactique d'interfaces spécialisées, celle-ci nous permettra de bien tester les exigences fonctionnelles de gestion de trajets selon un utilisateur, puisque ce service aura plus fonctions clées.

ADD-Limiter la complexité

Concept de design	Pour	Contre	Valeur	Cout
• Limiter la complexité de la structure	En réduisant la complexité du système, celui-ci devient plus facilement testable.	Si on limite la complexité, on limite d'autres aspects importants de la conception logiciel comme l'extensibilité et l'encapsulation.	2	1
• Limiter le non-déterminisme	Si on limite la complexité comportementale du système, comme le parallélisme non contraint, celui-ci devient plus facile à tester.	Difficile à cibler les comportements non déterministes et à les remplacer.	2	2

Quelle tactique avez vous choisi et pourquoi?

Nous avons choisi de limiter la complexité de la structure. Étant un service assez simple, le fait de limiter la complexité pour mieux tester ne nous pénalisera pas au niveau de la conception logiciel.

ADD-Usabilité

Identifiant	Description
CU01-U1	
CU02-U1	
CU03-U1	Pour le monitoring, la tactique maintain system model a été choisi.
CU04-U1	
CU05-U1	
CU06-U1	
CU07-U1	
CU08-U1	
CU09-U1	
CU10-U1	

ADD-Supporter l'initiative de l'usager

Concept de design	Pour	Contre	Valeur	Cout
• Cancel	Une commande peut-être annulé par l'utilisateur facilement, sans causer d'effets secondaires.	Un écouteur doit être implémenté pour réagir à l'annulation.	1	2
• Undo	Permet à l'utilisateur de retourner le système à un état ultérieur.	Un patron memento doit être implémenté et pas toutes les actions sont réversibles.	2	2
• Pause/resume	Permet de libérer temporairement les ressources, ces dernières peuvent être utilisées ailleurs.	Complexe à implémenter.	2	2
• Aggregate	Permet à l'utilisateur d'effectuer des tâches répetitives plus facilement.	Implémentation nécessaire.	2	1

Quelle tactique avez vous choisi et pourquoi?

Nous avons choisis la tactique aggregate. Puisque notre micro-service fera la gestion de trajets selon un utilisateur, il sera important de permettre à l'utilisateur de, par exemple, supprimer plusieurs trajets en même temps.

ADD-Supporter l'initiative du système

Concept de design	Pour	Contre	Valeur	Cout
• Maintain task model	Utile pour faciliter les tâches que devra réaliiser l'utilisateur, permet au système d'avoir une certaine connaissance du contexte des tâches et de les optimiser.	C'est un travail minutieux à faire pour s'assurer que chaque amélioration fonctionne bien, facile d'en oublier.	2	2
• Maintain user model	Utile pour connaître les habitudes de certains types d'utilisateurs et permet de s'adapter à ces utilisateurs en changeant certaines fonctionnalités ou en permettant de les changer.	Il peut être difficile de garder un modèle de l'utilisateur dans un système et de permettre une flexibilité selon le type.	2	2

Concept de design	Pour	Contre	Valeur	Cout
• Maintain system model	Utile pour montrer à l'utilisateur que le système travaille même s'il ne donne pas de résultats immédiats.	L'état du système doit être représenté dans la vue et ce de façon évidente, peut prendre un peu de temps à implémenter.	2	2

Quelle tactique avez vous choisi et pourquoi?

Nous avons choisis la tactique maintain system model, il est facile de bien représenter l'état des autres micro-services dans un dashboard. De plus, l'utilisateur aura facilement accès à ce tableau et pourra lui-même vérifier l'état des services.

ADD-Interopérabilité

Identifiant	Description
CU01-I1	
CU02-I1	
CU03-I1	Pour le monitoring, nous avons choisi la tactique discover service.
CU04-I1	
CU05-I1	
CU06-I1	
CU07-I1	
CU08-I1	
CU09-I1	
CU10-I1	

ADD-Localiser

Concept de design	Pour	Contre	Valeur	Cout

Concept de design	Pour	Contre	Valeur	Cout
• Discover service	Centralise l'information pour communiquer avec n'importe quelle service. Pas besoin de connaître personnellement le service qu'on veut rejoindre. De plus, lorsqu'un service change ses informations il suffit de demander au service de discovery.	Puisque c'est centralisé, il est très important que le service de Discovery soit fiable et ne soit pas susceptible de tomber en panne facilement.	3	2

Quelle tactique avez vous choisi et pourquoi?

Discover service, elle permet de centraliser les informations des micro-services dans une architecture. Pas besoin de connaître les nouveaux services dans un système, il suffit de communiquer avec le service de discovery.

ADD-Gérer les interfaces

Concept de design	Pour	Contre	Valeur	Cout
• Orchestrate	Permet aux systèmes d'intéragir plus facilement lorsque les interactions sont complexes.	Difficile à implémenter, nécessite l'implémentation d'un patron, par exemple, médiateur.	2	2
• Tailor interface	Permet de cacher des fonctions particulières aux utilisateurs non connus.	Peut enlever des fonctions importantes de l'interface.	2	1

Quelle tactique avez vous choisi et pourquoi?

Nous avons choisis la tactique Tailor interfaces. Pour notre micro-service de gestion de trajets selon un utilisateur, nous permettront l'accès à certaines fonctions seulement aux utilisateurs connus.

Réalisation des cas d'utilisation

RDCU-CU01 - Veux comparer les temps de trajet.

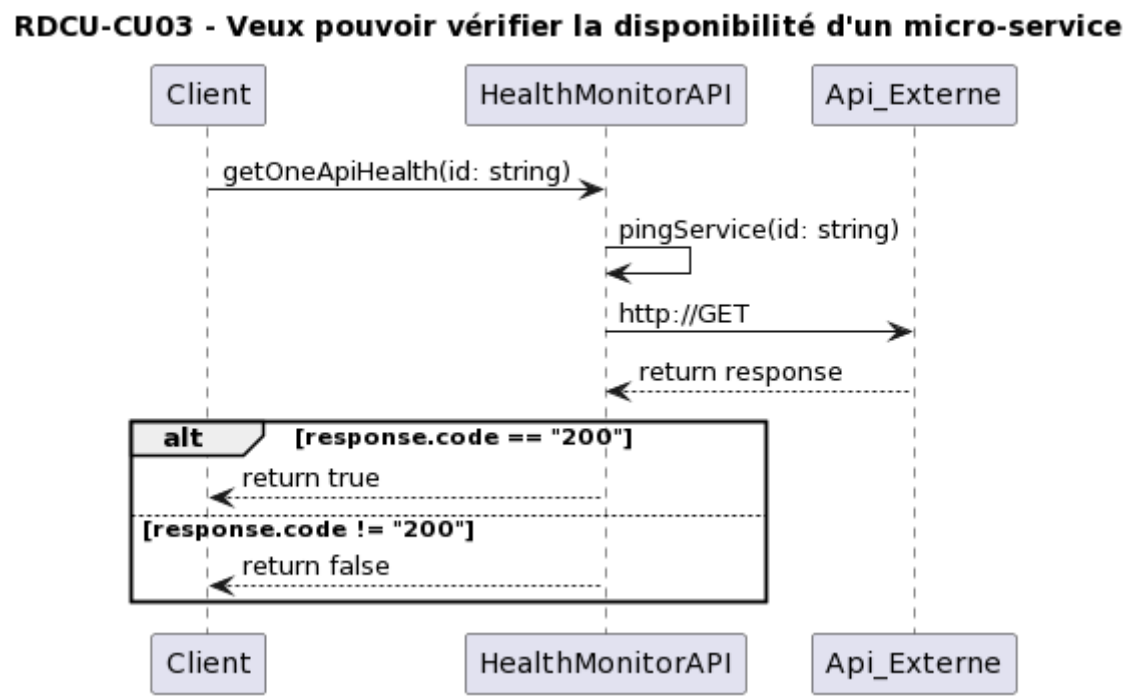
Diagramme(s) de séquence démontrant la réalisation de ce cas d'utilisation

RDCU-CU02 - Veux pouvoir mettre le chaos dans les

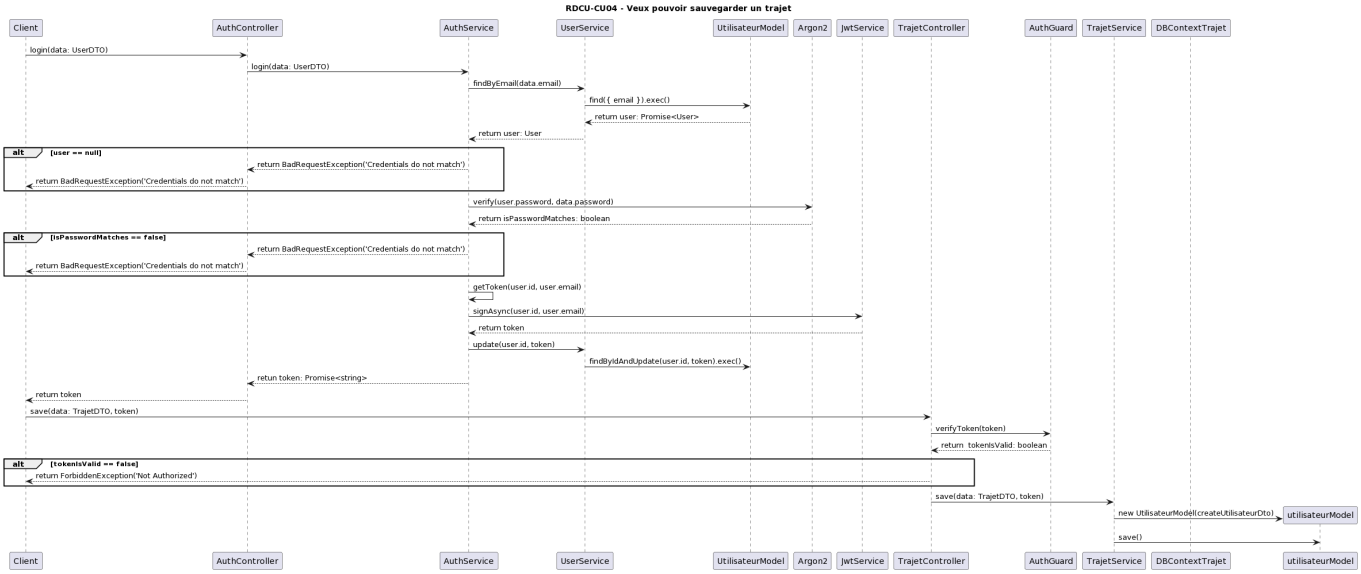
services en mode.

Diagramme(s) de séquence démontrant la réalisation de ce cas d'utilisation

RDCU-CU03 - Veux pouvoir vérifier la disponibilité d'un micro-service



RDCU-CU04 - Veux pouvoir sauvegarder un trajet



RDCU-CU05 -

Diagramme(s) de séquence démontrant la réalisation de ce cas d'utilisation

RDCU-CU06 -

Diagramme(s) de séquence démontrant la réalisation de ce cas d'utilisation

RDCU-CU07 -

Diagramme(s) de séquence démontrant la réalisation de ce cas d'utilisation

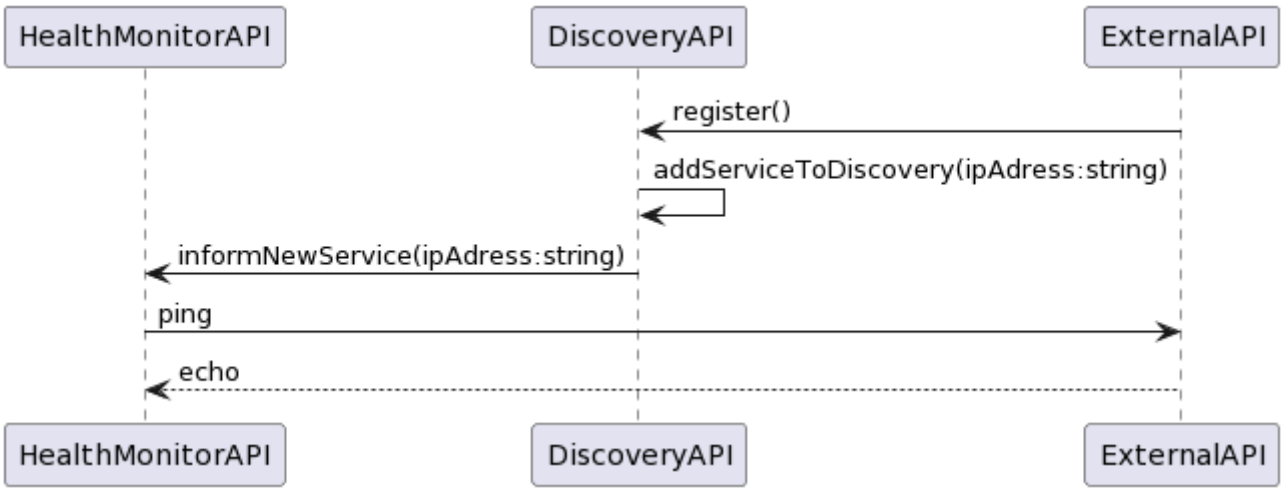
RDCU-CU08 -

Diagramme(s) de séquence démontrant la réalisation de ce cas d'utilisation

RDCU-CU09 -

Diagramme(s) de séquence démontrant la réalisation de ce cas d'utilisation

RDCU-CU10 -

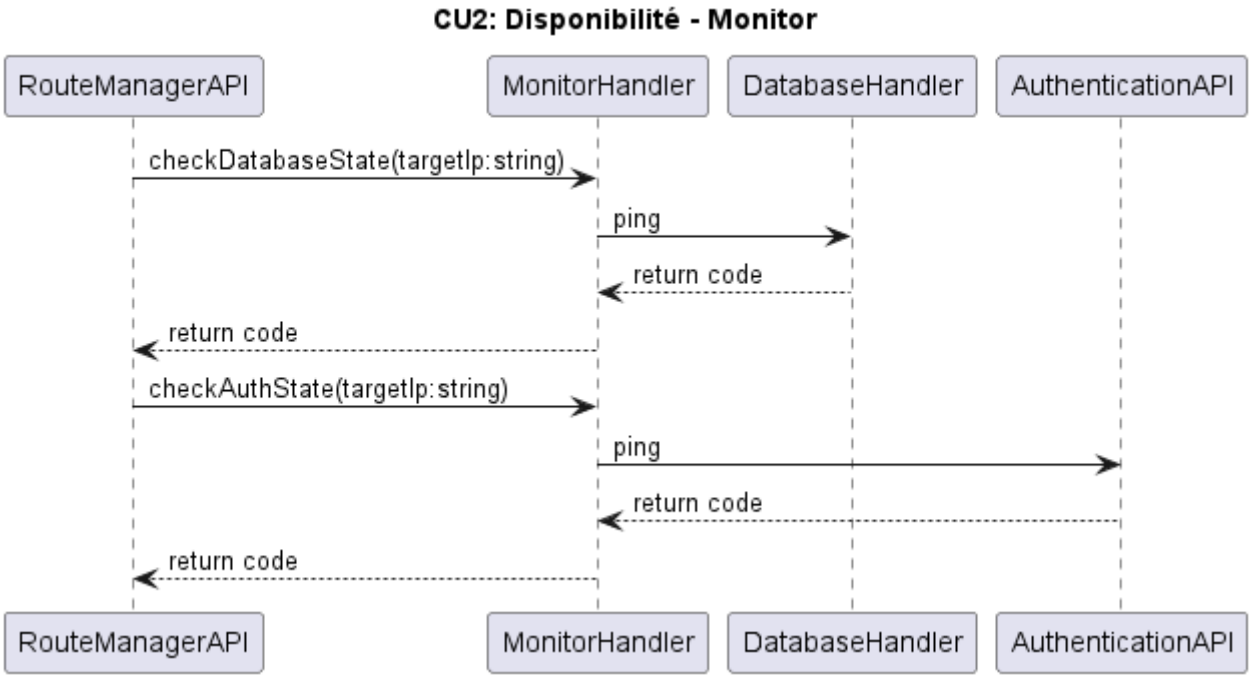


Réalisation des attributs de qualité

RDAQ-Disponibilité

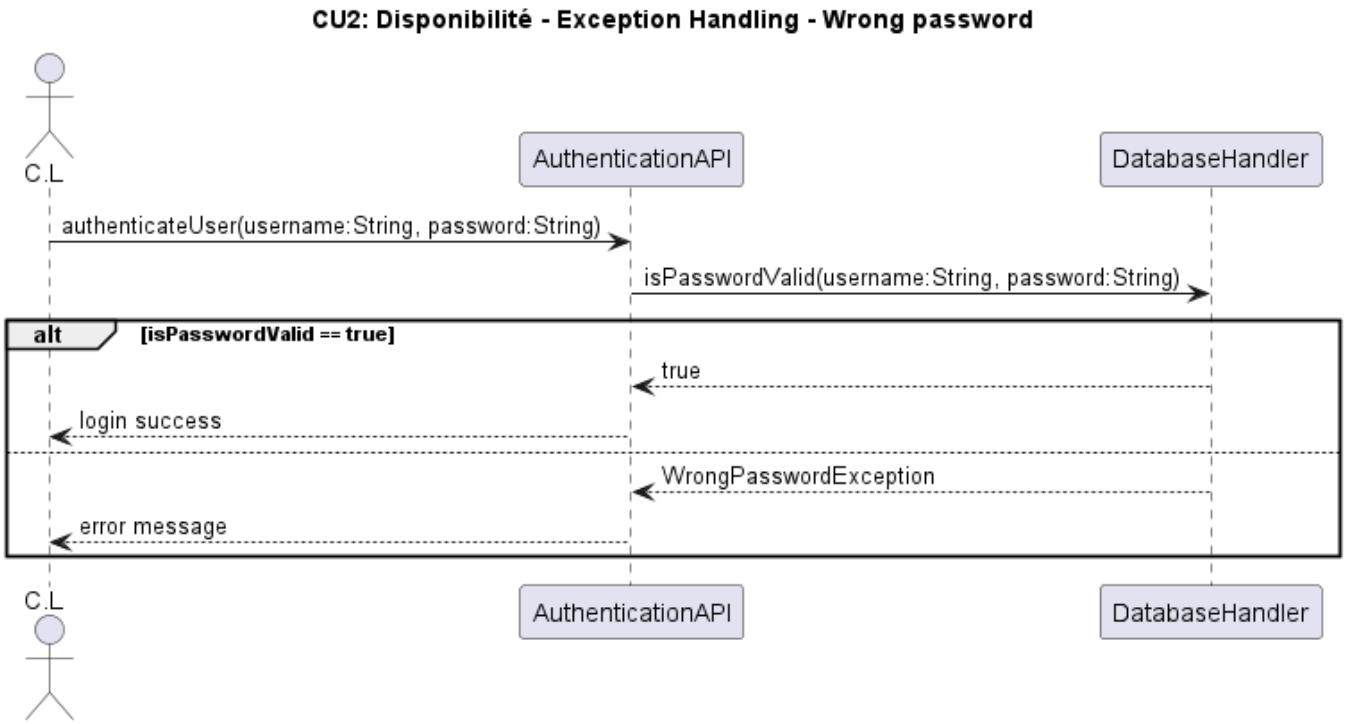
RDTQ-Détection de faute

Monitor



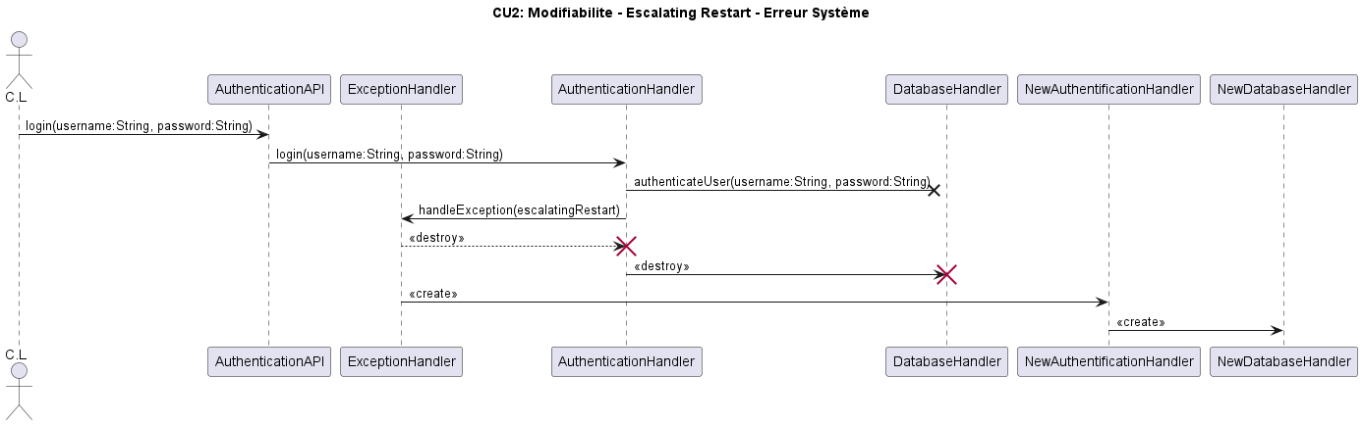
RDTQ-Préparation et réparation

Exception Handling



RDTQ-Réintroduction

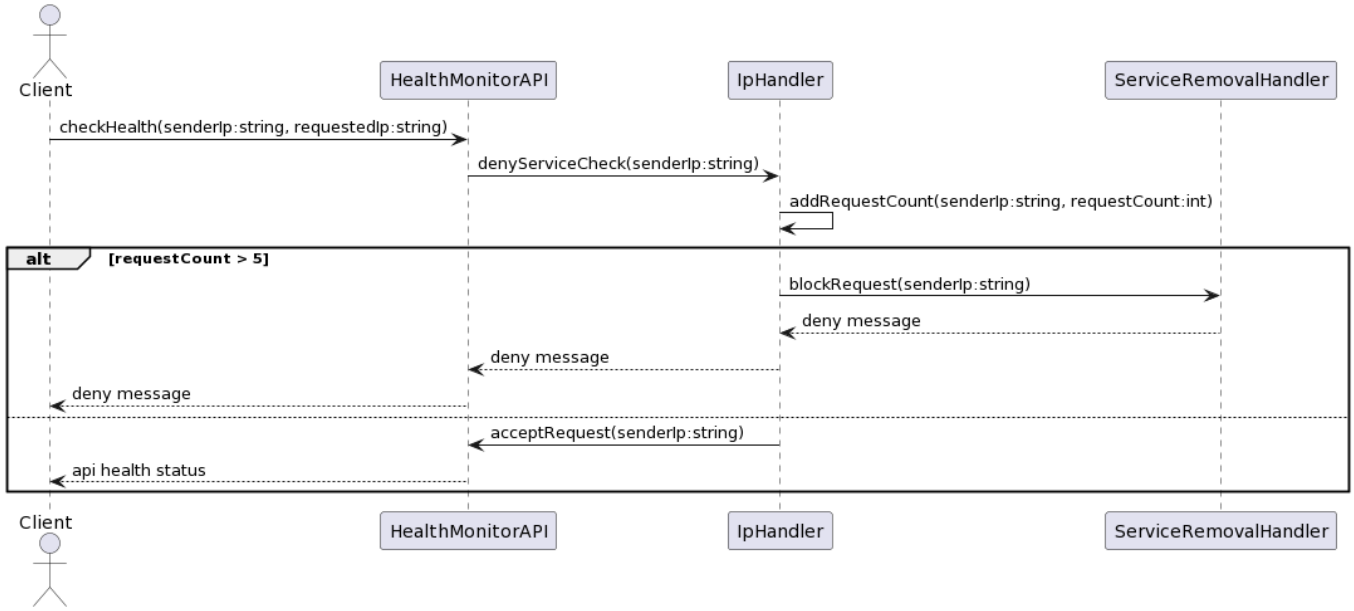
Escalating Restart



Dans ce diagramme de RDCU, on utilise la tactique d'escalating restart. Ainsi, l'on détruit une instance de Authentication Handler et de DatabaseHandler, puis l'on en créer une nouvelle.

RDTQ-Prévention des fautes

Removal From Service



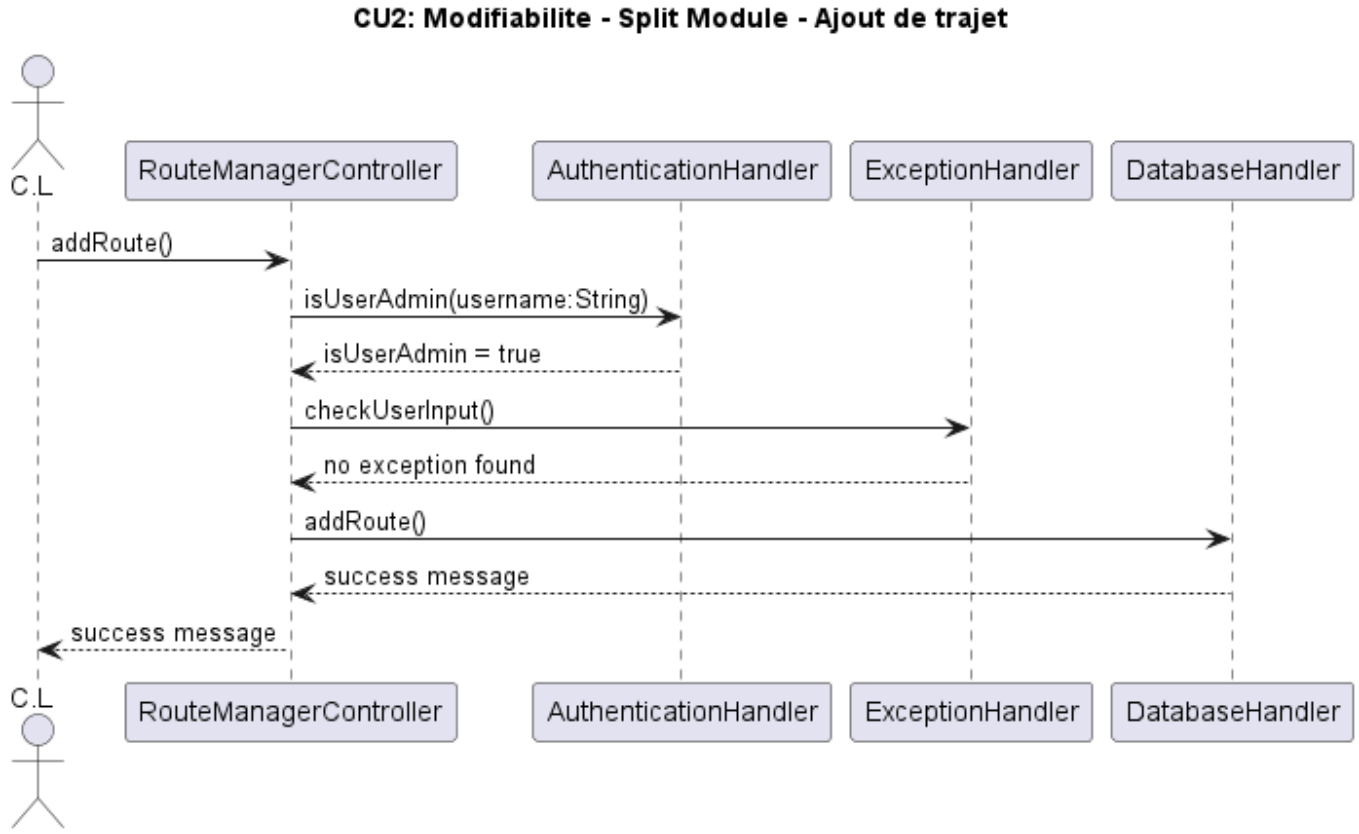
Relation entre les éléments architectuale et les exigences de disponibilité

Identifiant	Éléments	Description de la responabilité
CU01-D1		
CU02-D1		
CU03-D1		
CU04-D1		
CU05-D1		
CU06-D1		
CU07-D1		
CU08-D1		
CU09-D1		
CU10-D1		

RDAQ-Modifiabilité

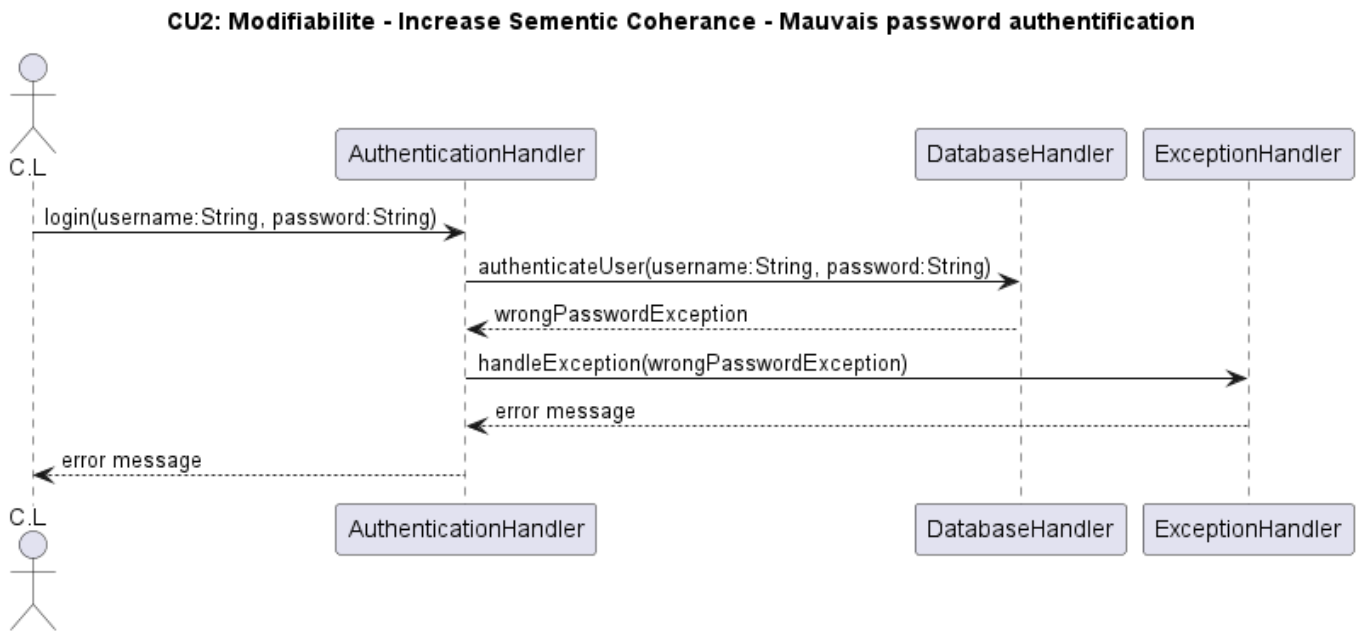
RDTQ-Réduire la taille des modules

Split Module



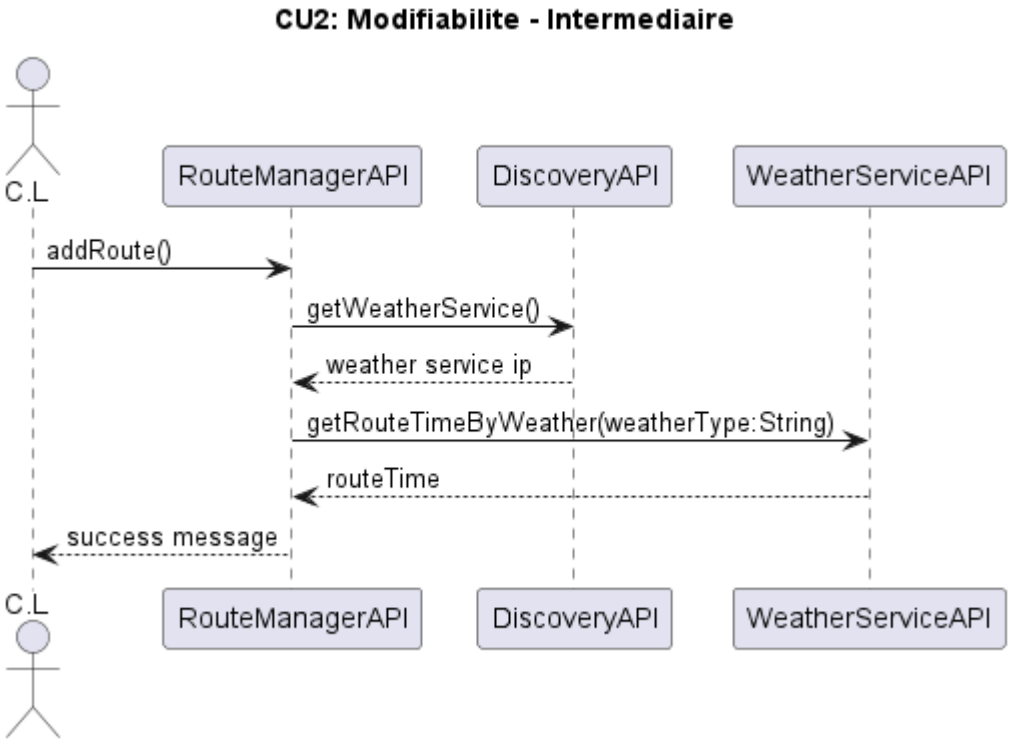
RDTQ-Augmenter la cohésion

Increase Semantic Coherance



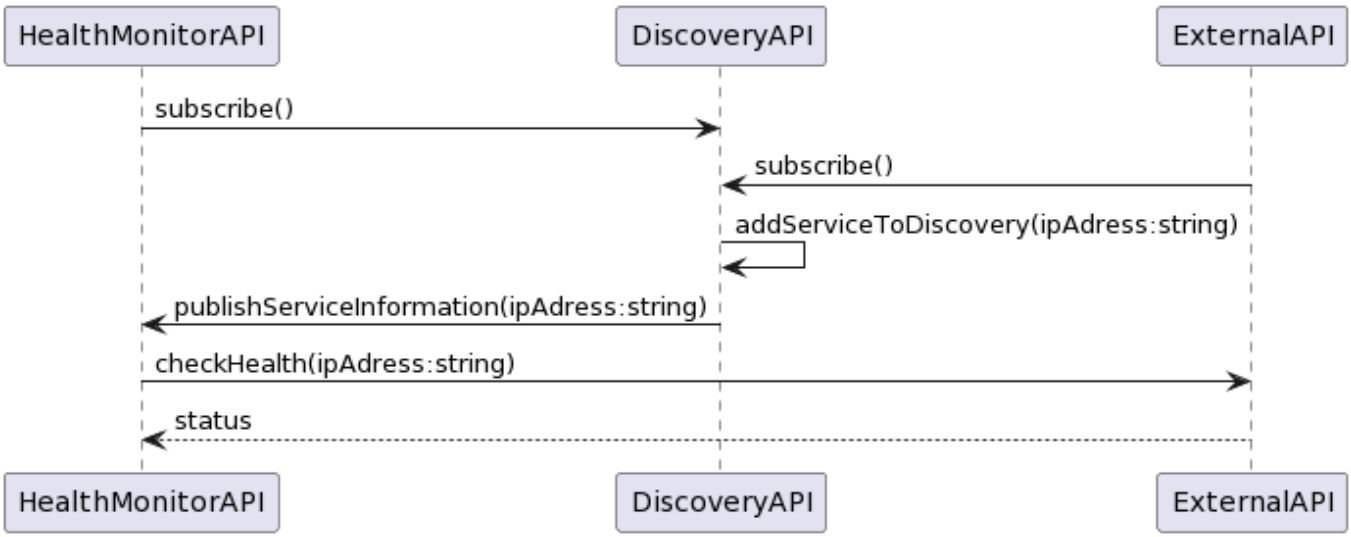
RDTQ-Réduire le couplage

Intermediaire



RDTQ-Defer binding

Defer Binding Time (with Publish-Subscribe)



Relation entre les éléments architectuale et les exigences de disponibilité

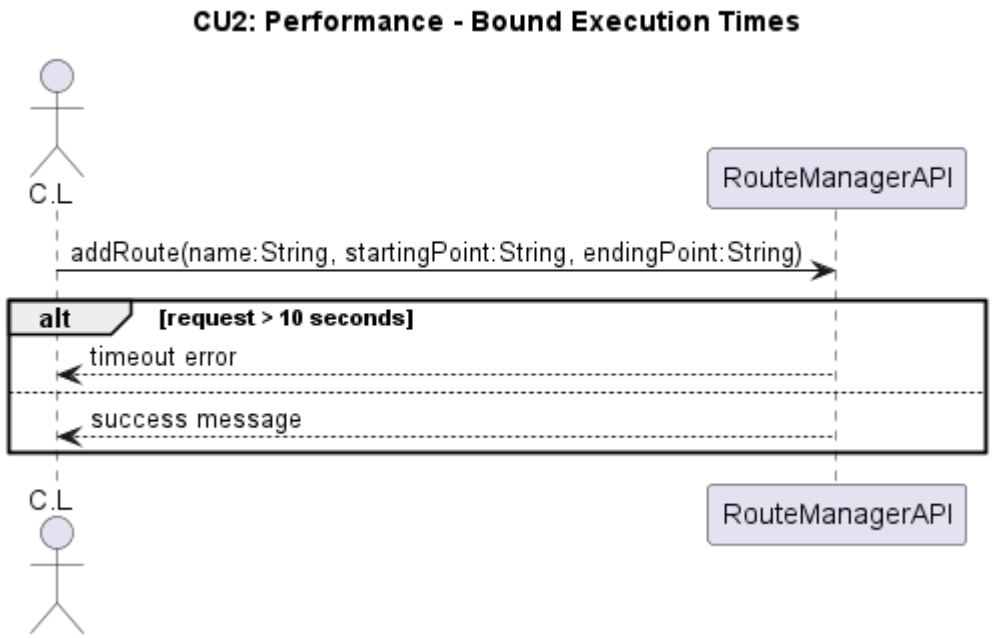
Identifiant	Éléments	Description de la responabilité
CU01-M1		
CU02-M1		
CU03-M1		
CU04-M1		
CU05-M1		

Identifiant	Éléments	Description de la responabilité
CU06-M1		
CU07-M1		
CU08-M1		
CU09-M1		
CU10-M1		

RDAQ-Performance

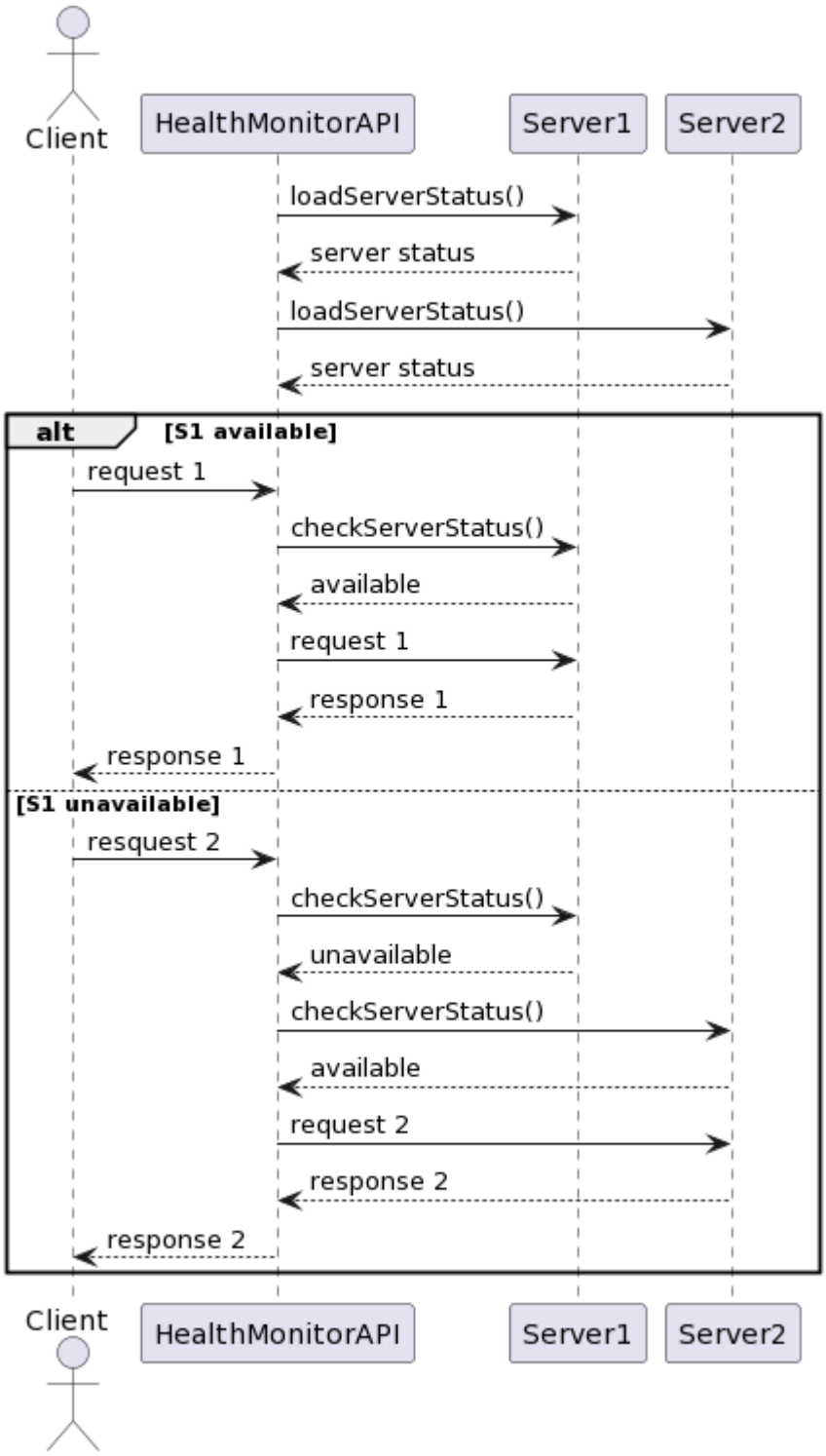
RDTQ-Contrôler la demande en ressources

Bound Execution Times



RDTQ-Gérer les ressources

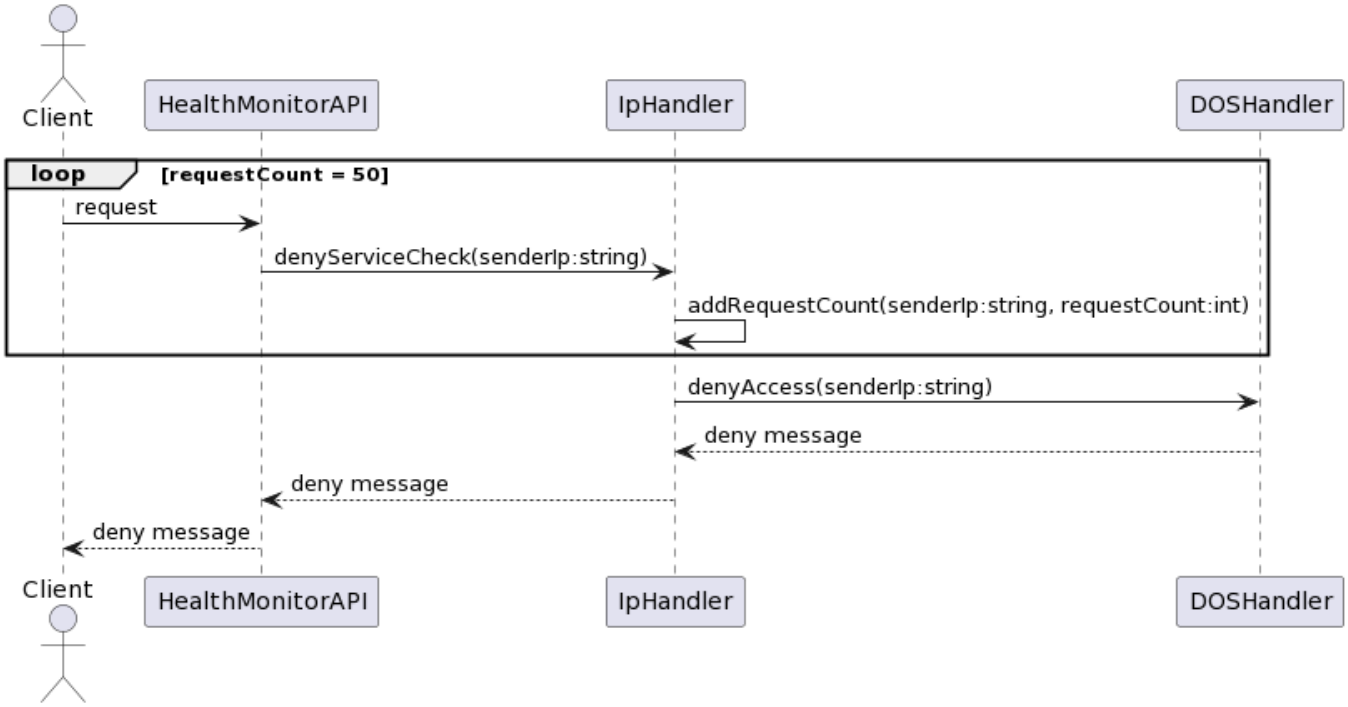
Maintain multiple copies of computation



RDAQ-Sécurité

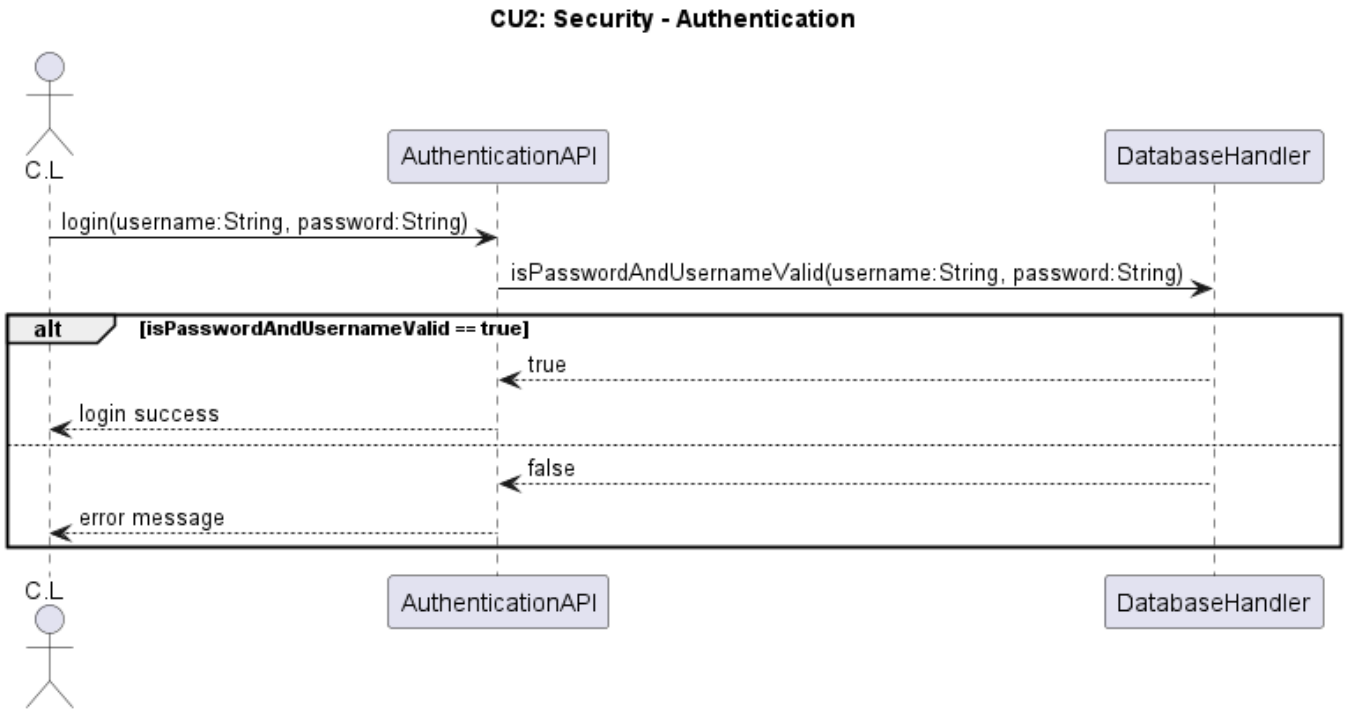
RDTQ-Détecter les attaques

Detect Servie Denial



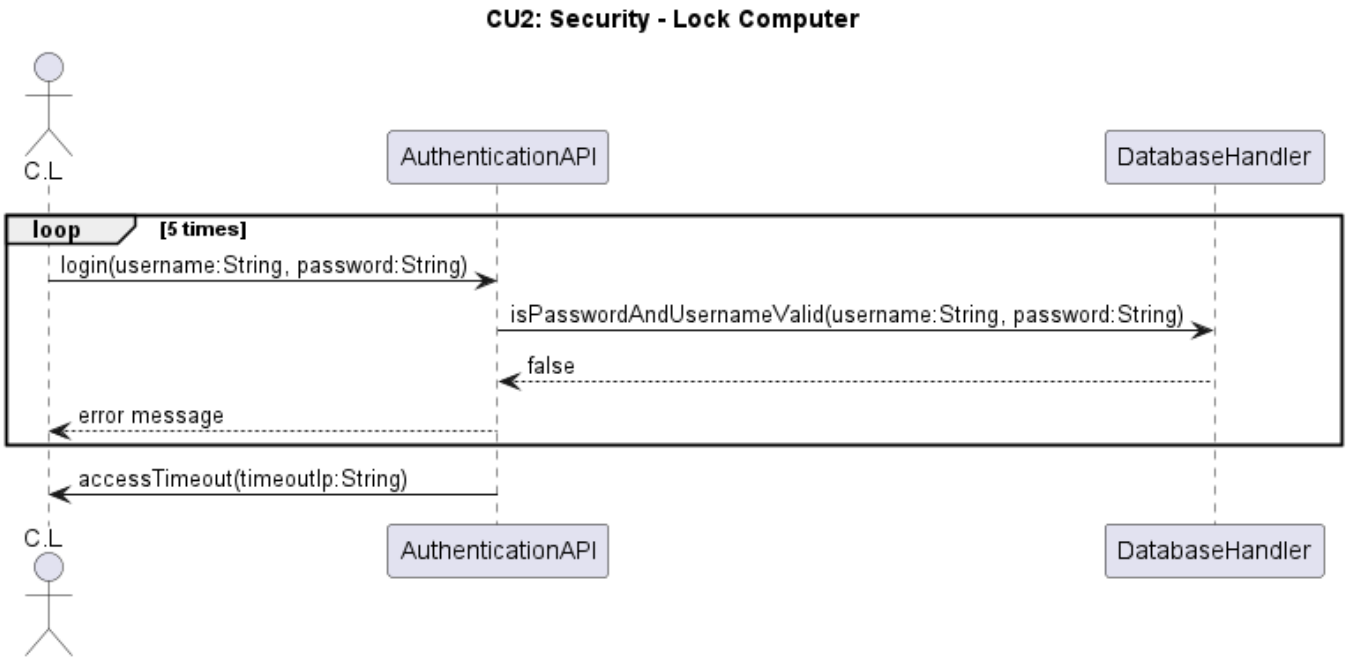
RDTQ-Résister aux attaques

Authentication



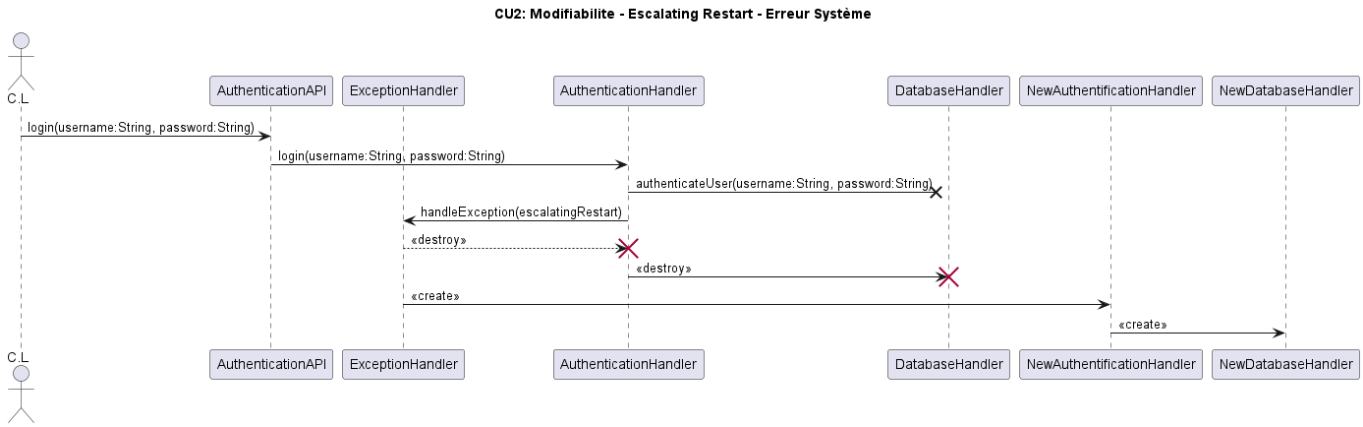
RDTQ-Réagir aux attaques

Lock computer



RDTQ-Récupérer d'une attaque

Escalating Restart (Restore)



Relation entre les éléments architectuale et les exigences de sécurité

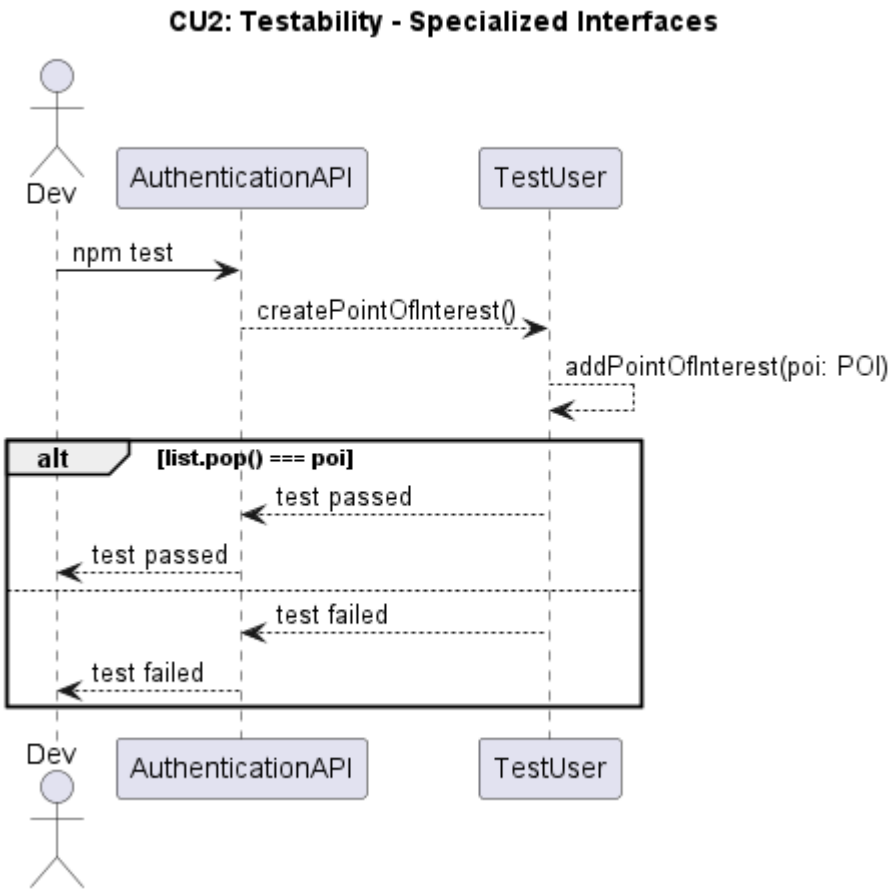
Identifiant	Éléments	Description de la responabilité
CU01-P1		
CU02-P1		
CU03-P1		
CU04-P1		
CU05-P1		
CU06-P1		
CU07-P1		
CU08-P1		

Identifiant	Éléments	Description de la responabilité
CU09-P1		
CU10-P1		

RDAQ-Testabilité

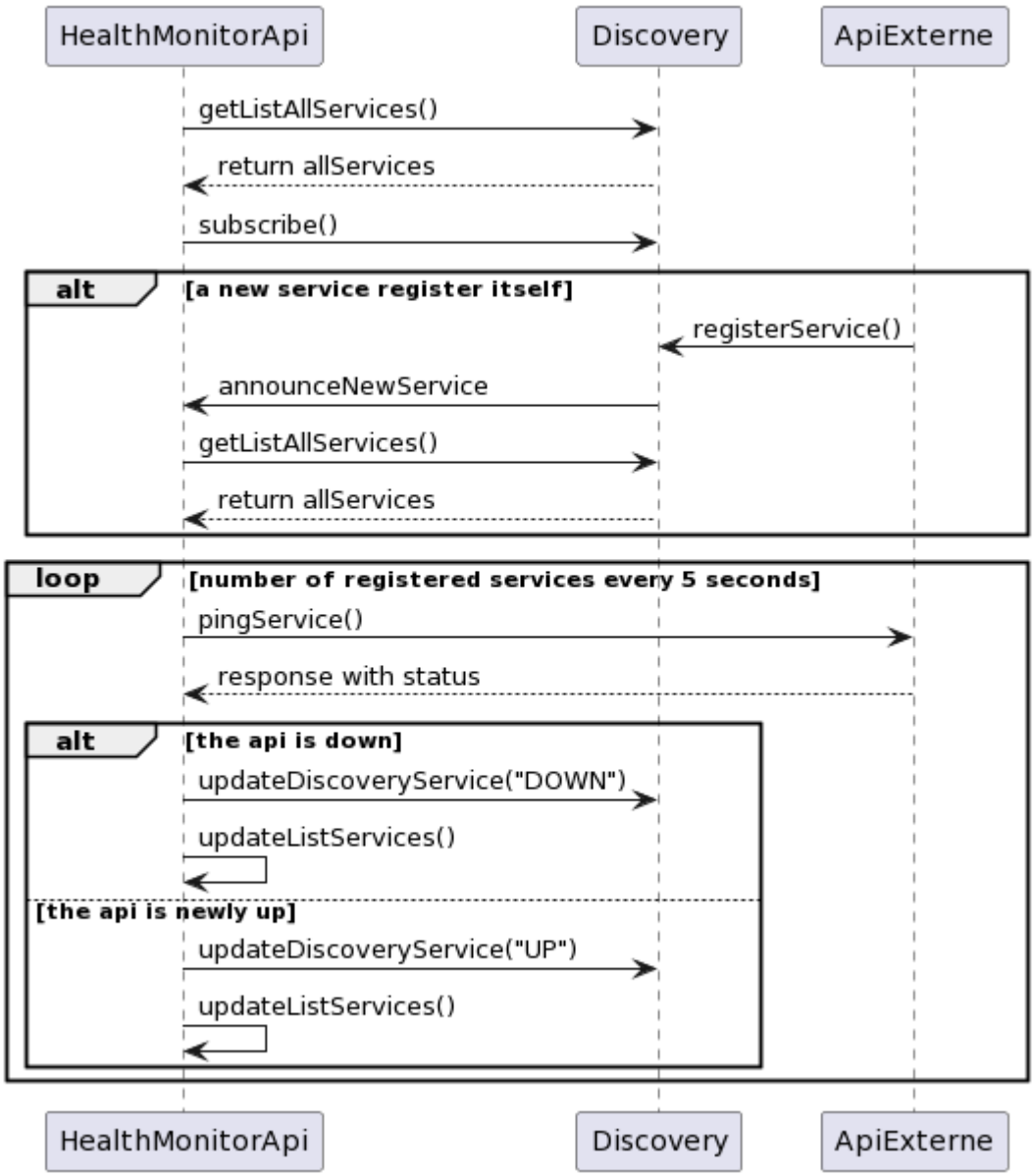
RDTQ-Contrôle et observe l'état du système

Specialized Interfaces



RDTQ-limiter la complexité

Limit Complexity



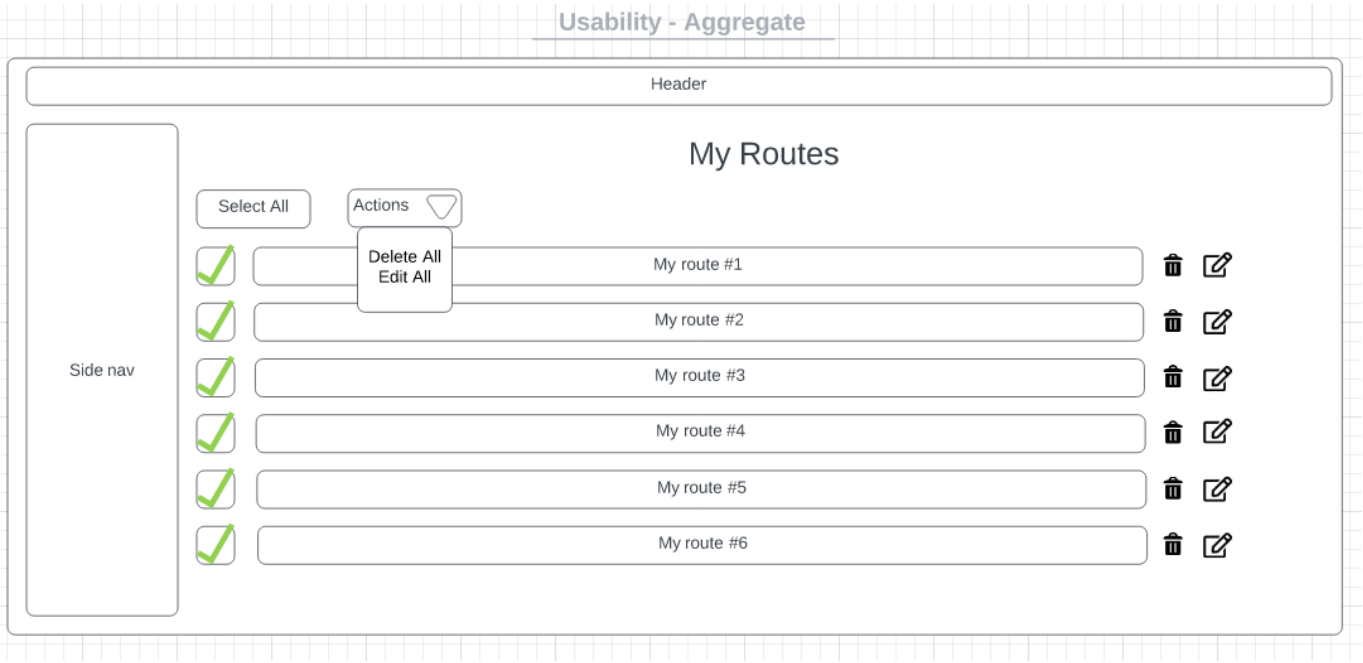
Relation entre les éléments architectuale et les exigences de testabilité

Identifiant	Éléments	Description de la responabilité
CU01-T1		
CU02-T1		
CU03-T1		
CU04-T1		
CU05-T1		
CU06-T1		
CU07-T1		
CU08-T1		
CU09-T1		
CU10-T1		

RDAQ-Usabilité

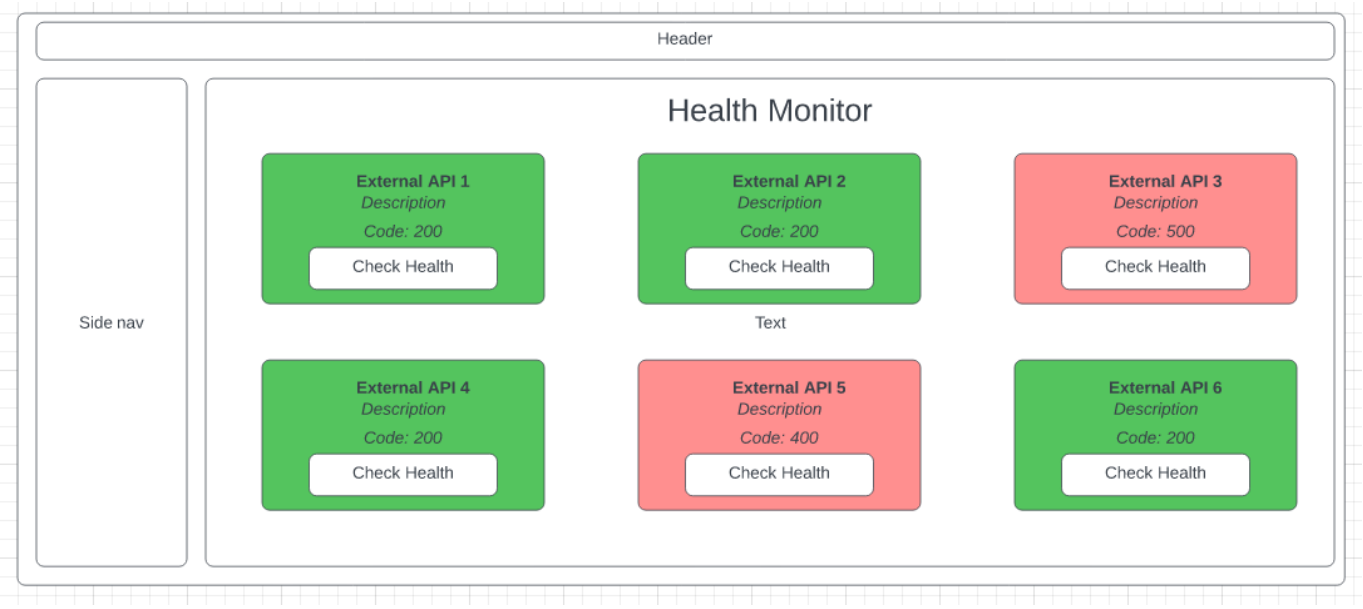
RDTQ-Supporter l'initiative de l'usager

Aggregate



RDTQ-Supporter l'initiative du système

Maintain System Model



Relation entre les éléments architectuale et les exigences d'usabilité

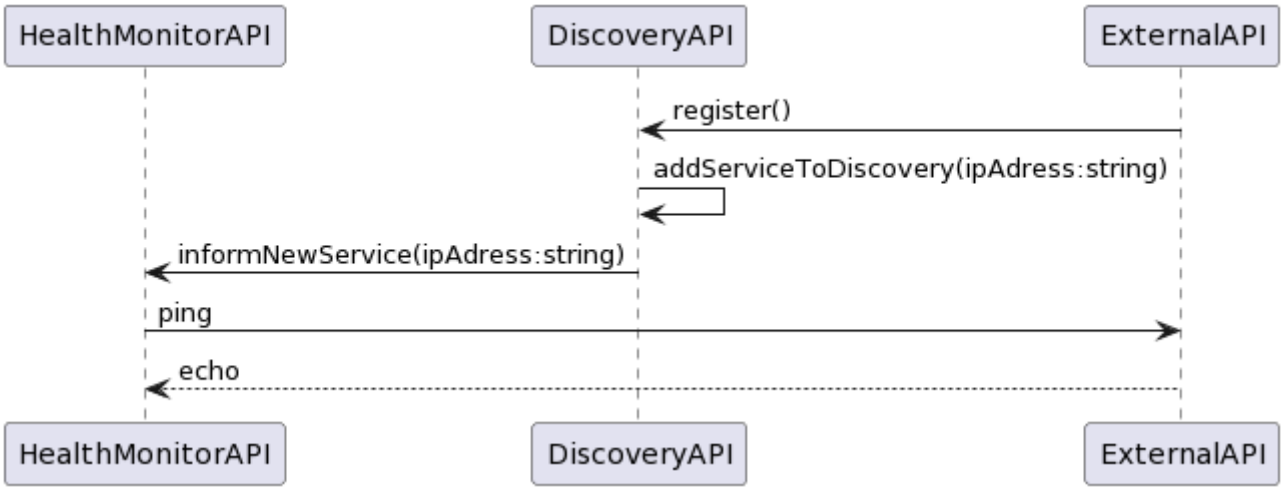
Identifiant	Éléments	Description de la responabilité
CU01-U1		
CU01-U2		

Identifiant	Éléments	Description de la responabilité
CU02-U1		
CU03-U1		
CU04-U1		
CU05-U1		
CU06-U1		
CU07-U1		
CU08-U1		
CU09-U1		
CU10-U1		

RDAQ-Interopérabilité

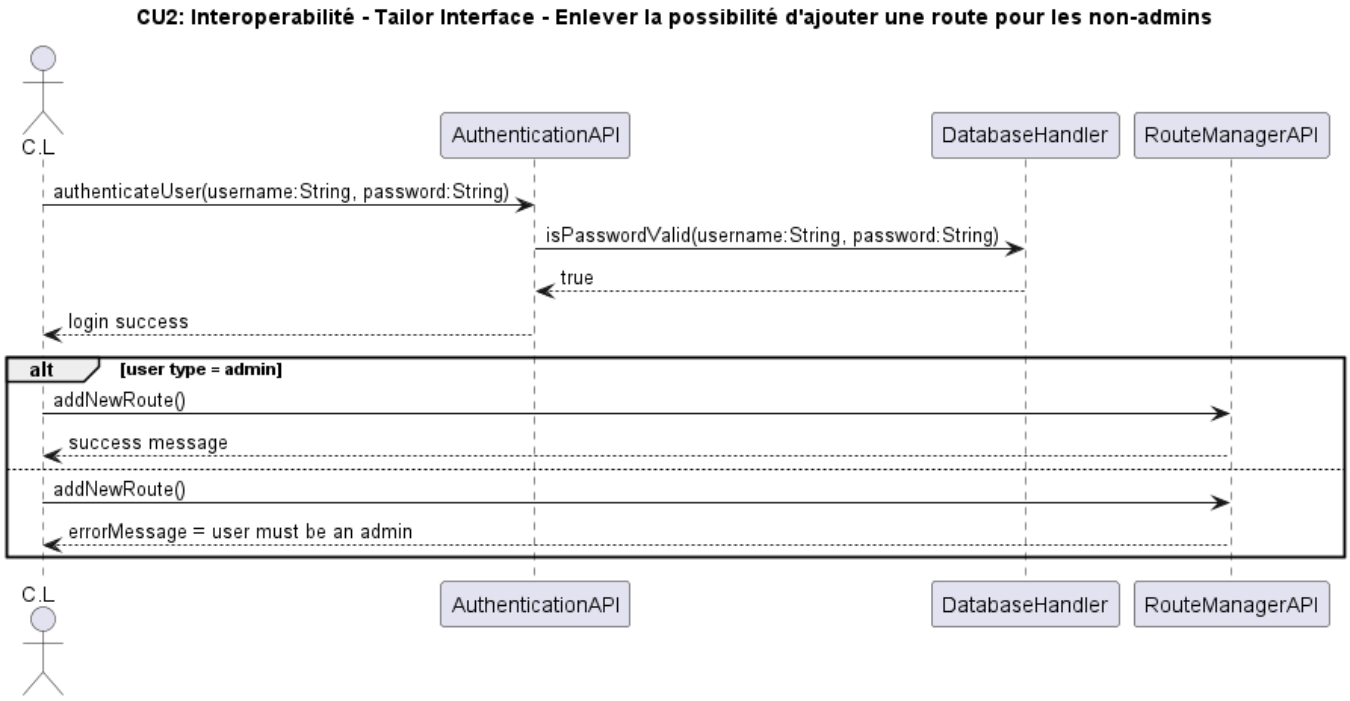
RDTQ-Localiser

Discover Service



RDTQ-Gérer les interfaces

Tailor interface



Relation entre les éléments architectuale et les exigences d'interopérabilité

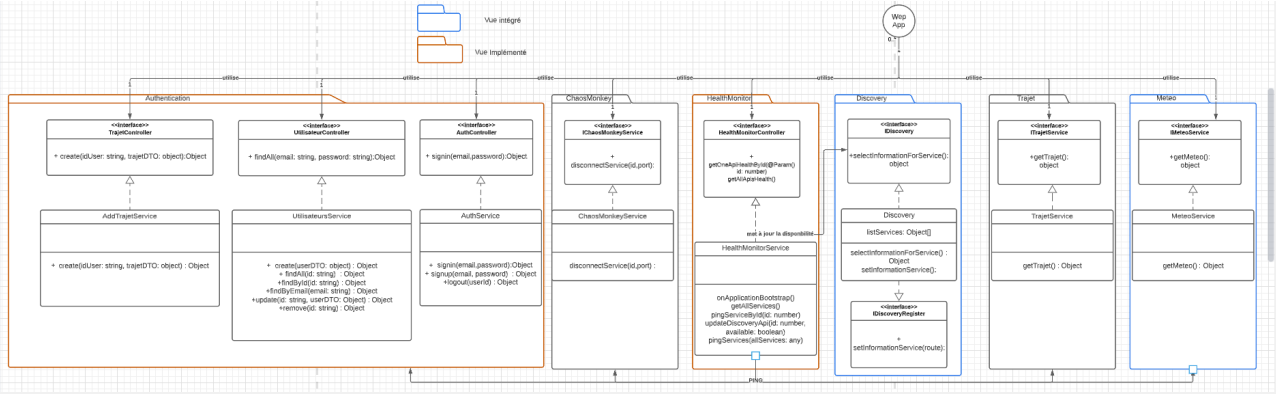
Identifiant	Éléments	Description de la responabilité
CU01-I1		
CU01-I2		
CU02-I1		
CU03-I1		
CU04-I1		
CU05-I1		
CU06-I1		
CU07-I1		
CU08-I1		
CU09-I1		
CU10-I1		

Vues architecturales

Vues architecturales de type Module

Vue #1

> **Présentation primaire**

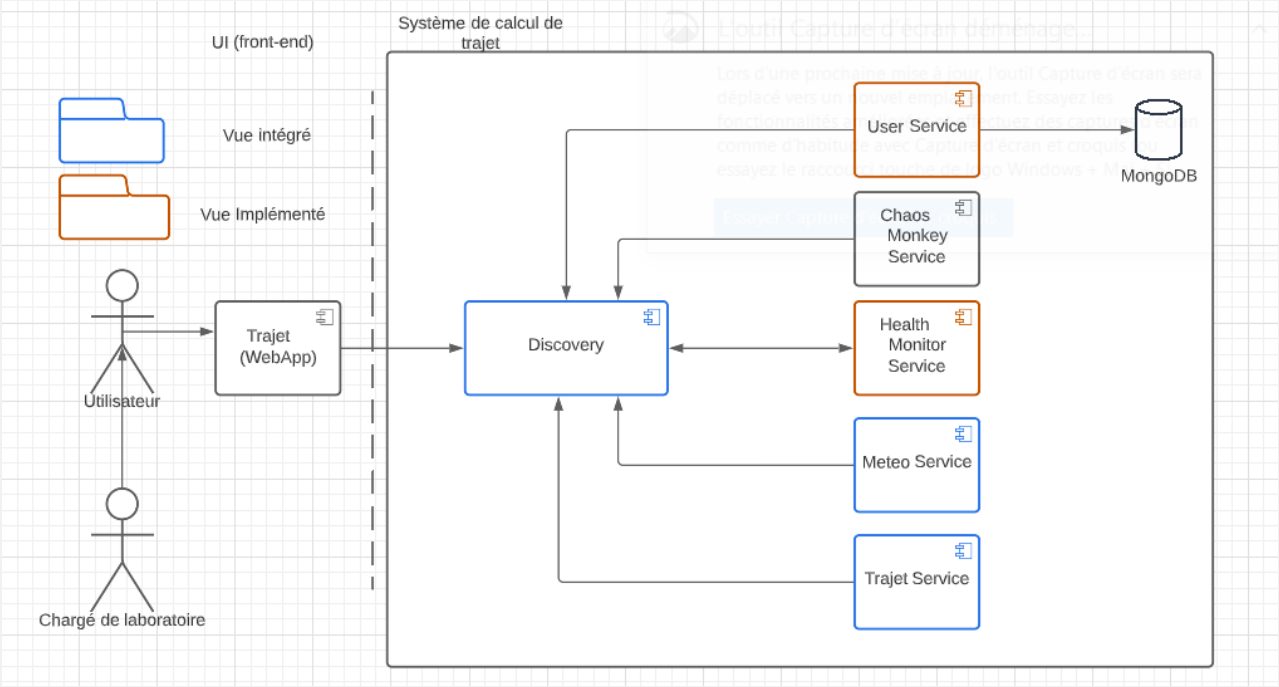


Catalogue d'éléments

Élement	Description	lien vers document d'interfaces
Discovery	Instance de l'objet Discovery. Ce module enregistre les nouveaux micro-services dans le système et envoie les informations pour contacter les différents micro-services.	
IDiscovery	Point d'entrée de Discovery.	
IDiscoveryRegister	Interface de Discovery qui permet de s'enregister au près du micro-service.	
HealthMonitorService	Instance de l'objet HealthMonitorService. Ce module envoie des requêtes PING aux autres modules du système afin de vérifier leurs état.	
HealthMonitorController	Point d'entrée du Health Monitor qui permet aux autres micro-services de communiquer au module via une requête HTTP.	
TrajetService	Instance de l'objet TrajetService. Ce module permet de calculer le meilleur itinéraire pour un trajet donné.	
ITrajetService	Point d'entrée de TrajetService.	
MeteoService	Instance de l'objet MeteoService. Ce module permet de retourner la météo à un moment donné.	
IMeteoService	Point d'entrée de MeteoService.	
ChaosMonkeyService	Instance de l'objet ChaosMonkeyService. Ce module permet de déconnecter un micro-service du système afin de vérifier sa solidité.	
IChaosMonkeyService	Point d'entrée de ChaosMonkeyService.	
AuthService	Instance de l'objet AuthService. Ce module permet à l'utilisateur de créer un compte ainsi que de se connecter et se déconnecter.	

Élement	Description	lien vers document d'interfaces
UtilisateursService	Instance de l'objet UtilisateursService. Ce module agit comme un DAO auprès de la base de donnée.	
AddTrajetService	Instance de l'objet AddTrajetService. Ce module permet à l'utilisateur de sauvegarder un trajet.	
AuthController	Point d'entrée du AuthService qui permet aux autres micro-services de communiquer au module via une requête HTTP.	
UtilisateurController	Point d'entrée du UtilisateurService qui permet aux autres micro-services de communiquer au module via une requête HTTP.	
TrajetController	Point d'entrée du TrajetService qui permet aux autres micro-services de communiquer au module via une requête HTTP.	

Diagramme de contexte



Guide de variabilité

Raisonnement

Cette vue montre le fonctionnement du système de monitoring et de Authentication. Ainsi l'on peut voir que le module du HealthMonitorService envoie un PING aux autres modules afin de recevoir leurs état. Celui-ci met ensuite à jours Discovery afin que celui-ci partage l'état de services. Cela permet ainsi de détecter une faute dans le système et de réagir. Pour ce qui est du système d'authentication, celui-ci est subdivisé en trois modules. AuthService et AddTrajetService permettent de se connecter à son compte utilisateur et d'ajouter un

nouveau service au système via des points d'accès. UtilisateurService agit comme DAO et permet de connecter se connecter à la base de données tout en réduisant le couplage.

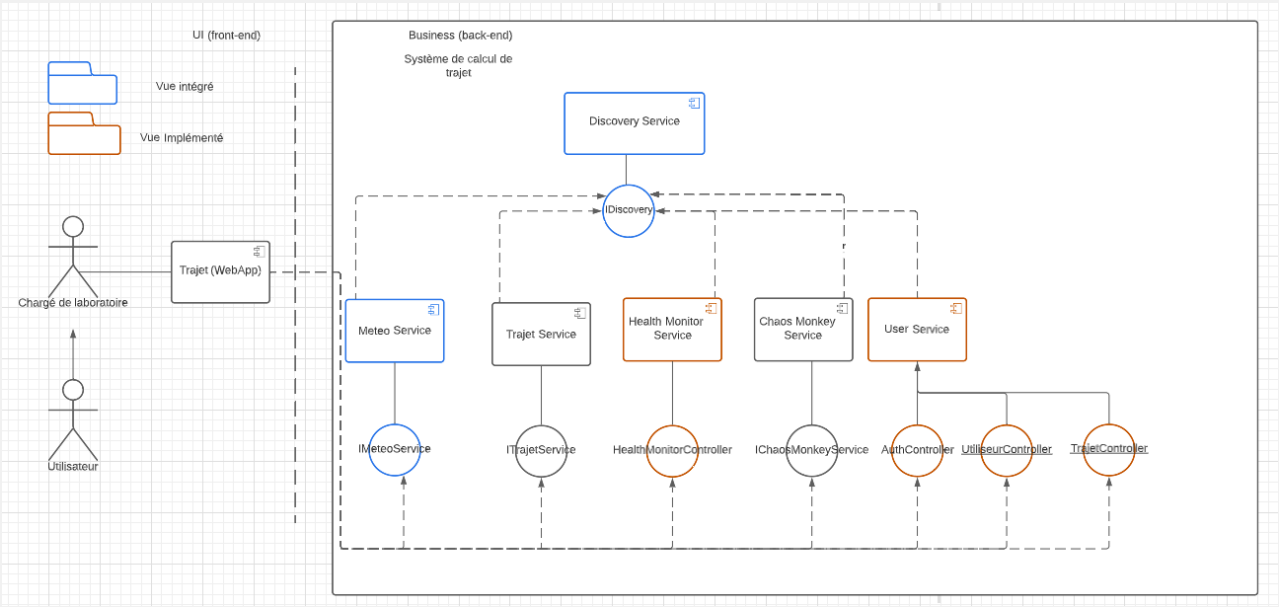
Vues associées

Vue #2...

Vues architecturales de type composant et connecteur

Vue #1

Présentation primaire

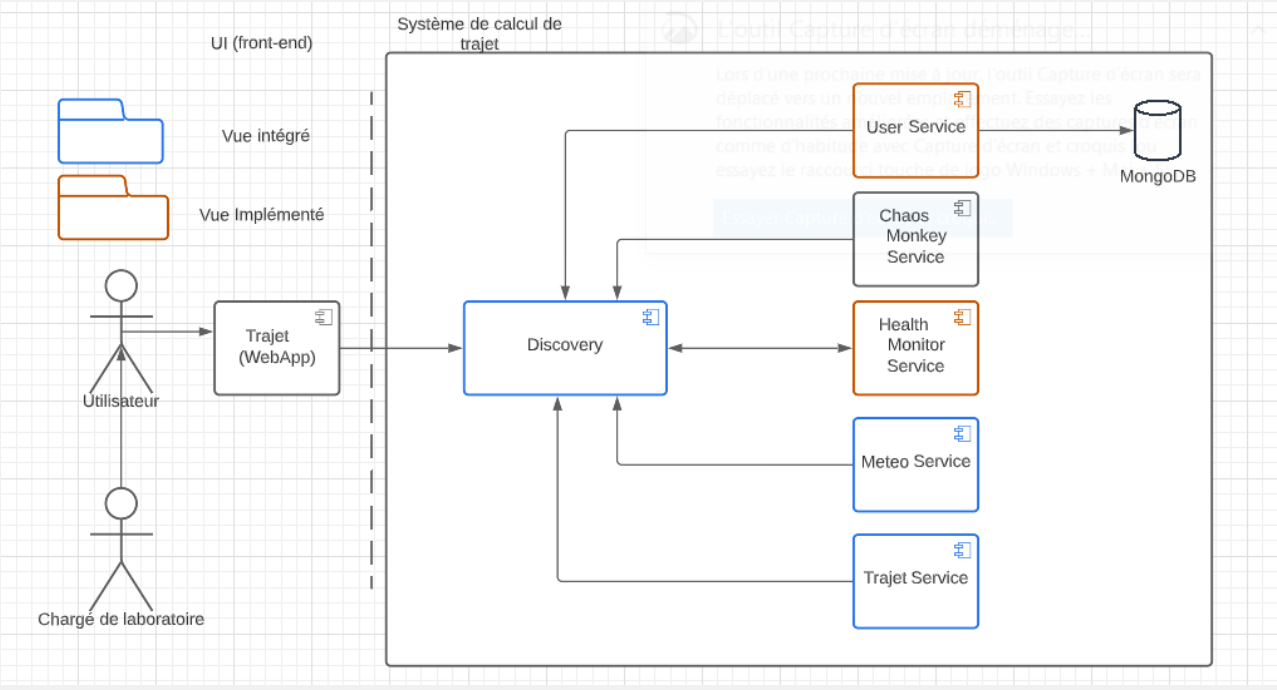


Catalogue d'éléments

Élement	Description	lien vers document d'interfaces
Discovery Service	Instance de Discovery. Ce composant enregistre les nouveaux micro-services dans le système et envoie les informations pour contacter les différents micro-services.	
IDiscovery	Point d'entrée de Discovery.	
HealthMonitorService	Composant HealthMonitorService. Ce composant permet de gérer le monitoring du système.	
HealthMonitorController	Point d'entrée du Health Monitor qui permet aux autres micro-services de communiquer au module via une requête HTTP.	
TrajetService	Composant TrajetService. Ce composant permet de calculer le meilleur itinéraire pour un trajet donné.	
ITrajetService	Point d'entrée de TrajetService.	

Élement	Description	lien vers document d'interfaces
MeteoService	Composant MeteoService. Ce composant permet de retourner la météo à un moment donné.	
IMeteoService	Point d'entrée de MeteoService.	
ChaosMonkeyService	Composant ChaosMonkeyService. Ce composant permet de déconnecter un micro-service du système afin de vérifier sa solidité.	
IChaosMonkeyService	Point d'entrée de ChaosMonkeyService.	
UserService	Composant UtilisateursService. Ce composant permet de gérer les comptes utilisateurs.	
AuthController	Point d'entrée du AuthService qui permet aux autres micro-services de communiquer au module via une requête HTTP.	
UtilisateurController	Point d'entrée du UtilisateurService qui permet aux autres micro-services de communiquer au module via une requête HTTP.	
TrajetController	Point d'entrée du TrajetService qui permet aux autres micro-services de communiquer au module via une requête HTTP.	

Diagramme de contexte



Guide de variabilité

Raisonnement

Cette vue permet de montrer les différents points d'accès des composants du système. Tout les micro-services utilisent l'interface de Discovery afin de s'enregistrer sur le service. Les services peuvent ensuite demander à Discovery les points d'accès des autres services pour y accéder. Pour ce qui est du health monitoring, un controleur permet d'indiquer quelle requête à été envoyé par les autre micro-services. Pour ce qui est de Authentication, trois controleur permettent de gérer les différentes actions de authentication. AuthController et TrajetController permettent respectivement de controler la connexion et déconnexion au système et de sauvegarder un nouveau trajet. UtilisateurController permet d'effectuer une facade entre la base de donnné et le système en agissant de point d'entrée au DAO qui est UtilisateurService.

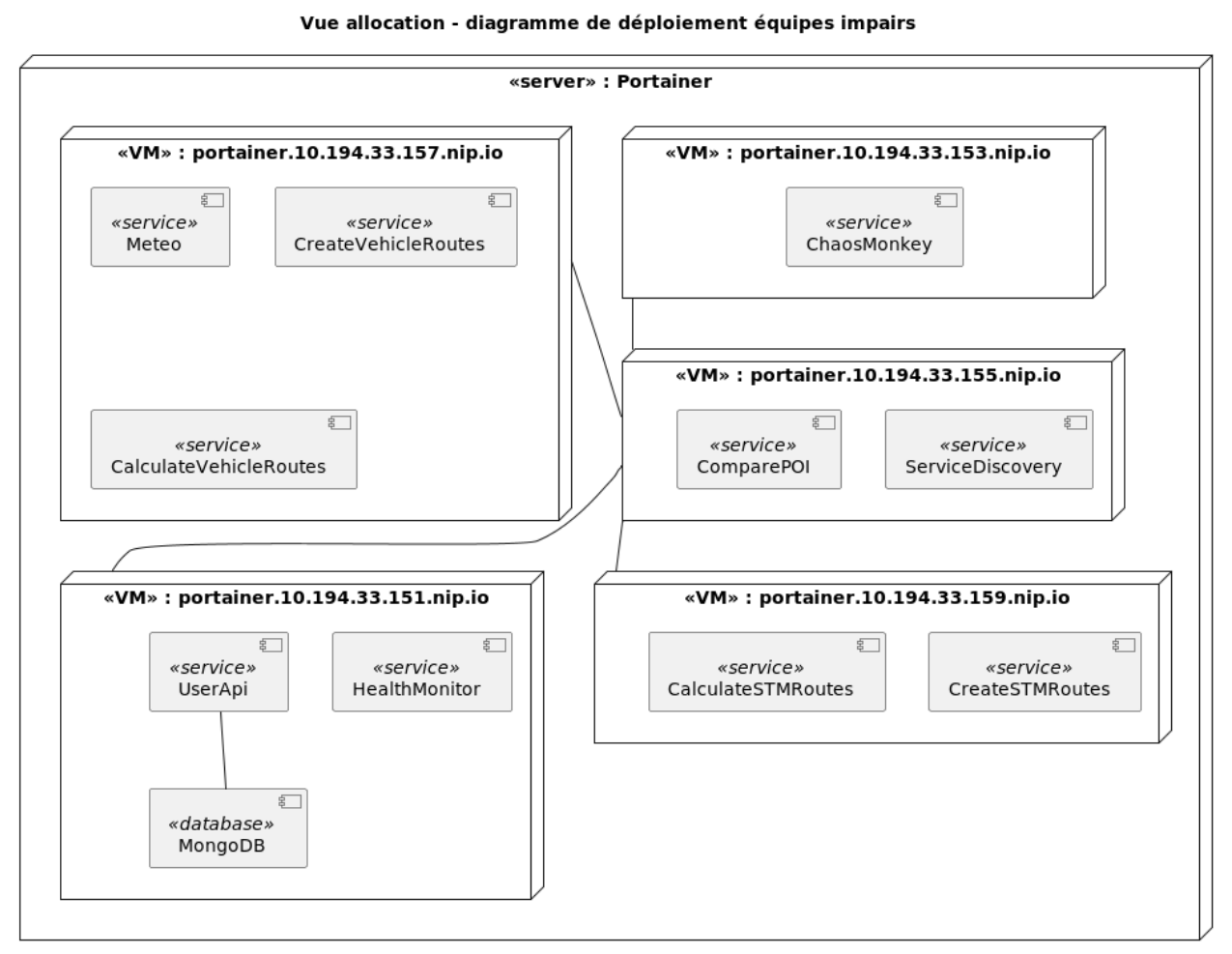
Vues associées

Vue #2...

Vues architecturales de type allocation

Vue #1

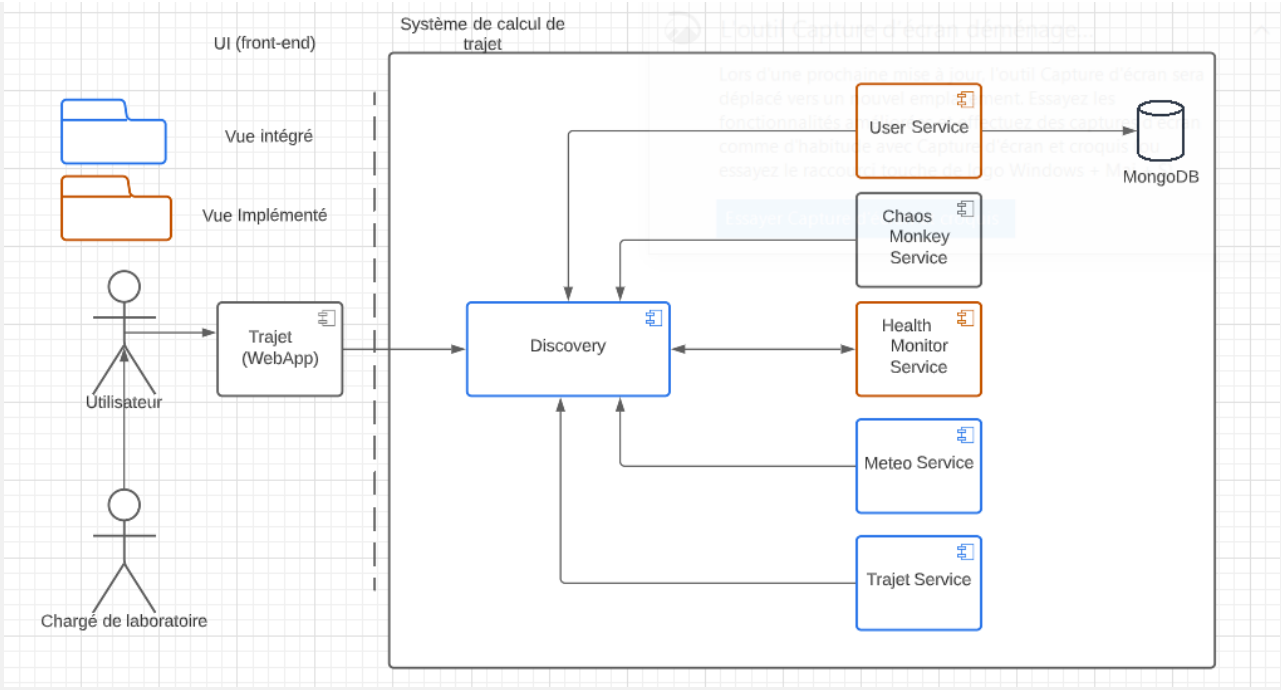
Présentation primaire



Catalogue d'éléments

Élement	Description	lien vers document d'interfaces
Meteo	Service meteo qui sert à renvoyer la météo pour un emplacement donnée.	
CreateVehiculeRoute	Service que l'utilisateur appel pour demander un trajet en voiture.	
CalculerVehiculeRoute	Ce service permet de calculer le temps pour un trajet en voiture envoyé par l'utilisateur. Ce service est complémentaire au service de CreateVehiculeRoute.	
ChaosMonkey	Ce service sert à déconnecter les autres micro-services du sytème afin de tester l'attribut de disponibilité de ceux-ci.	
ComparePOI	Ce service permet de comparer des points d'interêt marqué par l'utilisateur.	
ServiceDiscovery	Le service de Discovery permet de gérer l'accès aux différents services du système en envoyant les informations demander pour se connecter à un service.	
CalculateSTMRoutes	Ce service permet de calculer le temps pour un trajet en transport en commun envoyé par l'utilisateur. Ce service est complémentaire au service de CreateSTMRoutes.	
CreateSTMRoutes	Service que l'utilisateur appel pour demander un trajet en transport en commun.	
HealthMonitor	Ce service agit comme un monitor pour les autres services et permet de voir quels services sont actifs ou inactifs.	
UserAPI	Ce service sert à l'utilisateur afin de se connecter à son compte utilisateur et envoyé les trajets qu'il souhaite enregistrer dans la base de données.	
MongoBD	Base de donnée qui enregistre les trajets enregistrés des utilisateurs.	

> **Diagramme de contexte**



Guide de variabilité

Raisonnement

Ce diagramme divise les services dans plusieurs machines virtuelles qui appartiennent à chaque équipe. Tous les services sont reliés au service de Discovery afin de centraliser l'information des adresses pour accéder à chaque service. Le userAPI sert de porte d'accès pour la base de donnée qui contient les informations des trajets sauvegardés par le user.

Vues associées

Conclusion

En conclusion, nous avons explicité dans ce document les diagrammes et informations nécessaires afin de répondre aux objectifs d'affaires du laboratoire tout en implémentant les attributs de qualités. Ainsi le document d'architecture démontre au travers de vues modules, composants et connecteurs, déploiement et contexte la structure du système de trajet en intégrant les attributs de qualités au sein de ceux-ci.

Documentation des interfaces

Les catalogues d'élément devraient être des tableaux qui contiennent la description des éléments en plus d'un lien vers la documentation de l'interface de ceux-ci. Je vous suggère d'utiliser un document par interface pour vous faciliter la tâche. Il sera ainsi plus facile de distribuer la documentation d'une interface aux équipes en ayant besoin. La documentation des interfaces de vos éléments doit se faire en utilisant le gabarit suivant.

Voici quelques exemples de documentation d'interface utilisant ce gabarit:

- https://wiki.sei.cmu.edu/confluence/display/SAD/OpcPurchaseOrderService+Interface+Documentation
- https://wiki.sei.cmu.edu/confluence/display/SAD/OpcOrderTrackingService+Interface+Documentation

- https://wiki.sei.cmu.edu/confluence/display/SAD/WebServiceBroker+Interface+Documentation