Laboratoire #1

Groupe: 01

Equipe: 12

Membres de l'équipe: Maxime Lamontagne Morissette

Participation

> L'évaluation suivante est faite afin d'encourager des discussions au sein de l'équipe et afin de conserver une trace des réalisations de chacun, tout au long de la session. Une discussion saine du travail de chacun est utile afin d'améliorer le climat de travail. Les membres de l'équipe ont le droit de retirer le nom d'un ou une collègue du rapport en cas de non-participation.

> Une pénalité de 10% sera appliquée en cas de non-remplissage de cette section.

Nom de l'étudiant	Tâches réalisées	Facteur de participation
Maxime Lamontagne Morissette	Tout	1

Introduction /2

> L'objectif de ce laboratoire est d'explorer une architecture de microservices complexe qui communique avec des interfaces de programmation d'application externes et qui utilise des données réelle afin d'accomplir ses tâches. Le présent laboratoire permet aussi de mettre en pratique certaines tactiques d'attribut de qualité. Ce rapport présente un diagramme de séquence de haut niveau représentant le fonctionnement générale du système et des diagrammes de contexte de haut niveau concernant les différents conteneurs de computation. De plus, l'ajout d'une base de donnée au système est discuté dans le contexte d'amélioration du système et une tactique pour améliorer la disponibilité de l'application en cas d'attaque est proposée.

Diagramme de séquence de haut niveau

Explication générale du fonctionnement du système /2

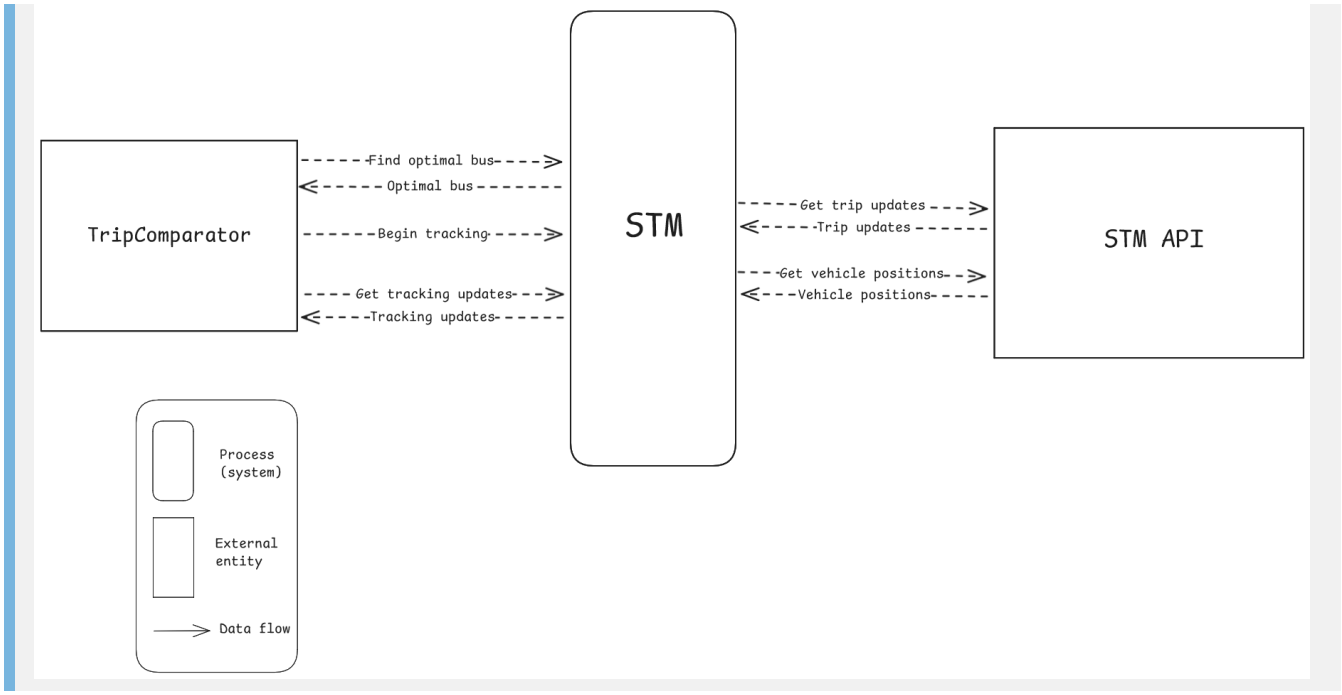
- **NodeController** : Le service NodeController permet la communication interservice. C'est ce service qui initie la comparaison de trajet et qui s'occupe de ralyer l'information
- **STM** : Le service STM permet de faire des appels à l'interface de programmation d'application du Service des Transport de Montréal. C'est ce service qui est responsable de fournir les données necéssaire pour trouver le meilleur autobus pour un trajet donné. Il s'occupe aussi de fournir les données en temps réel concernant la position de l'autobus, permettant ainsi au système de suivre l'autobus choisi.

- **RouteTimeProvider** : Le service RouteTimeProvider permet de faire des appels à l'interface de programmation d'application de TomTom afin d'obtnir des données concernant un trajet donné, fait en voiture.
- **TripComparator** : Le service TripComparator communique avec les services STM et RouteTimeProvider afin d'obtenir les données de l'autobus optimal pour un trajet et les données du même trajet, fait en voiture.

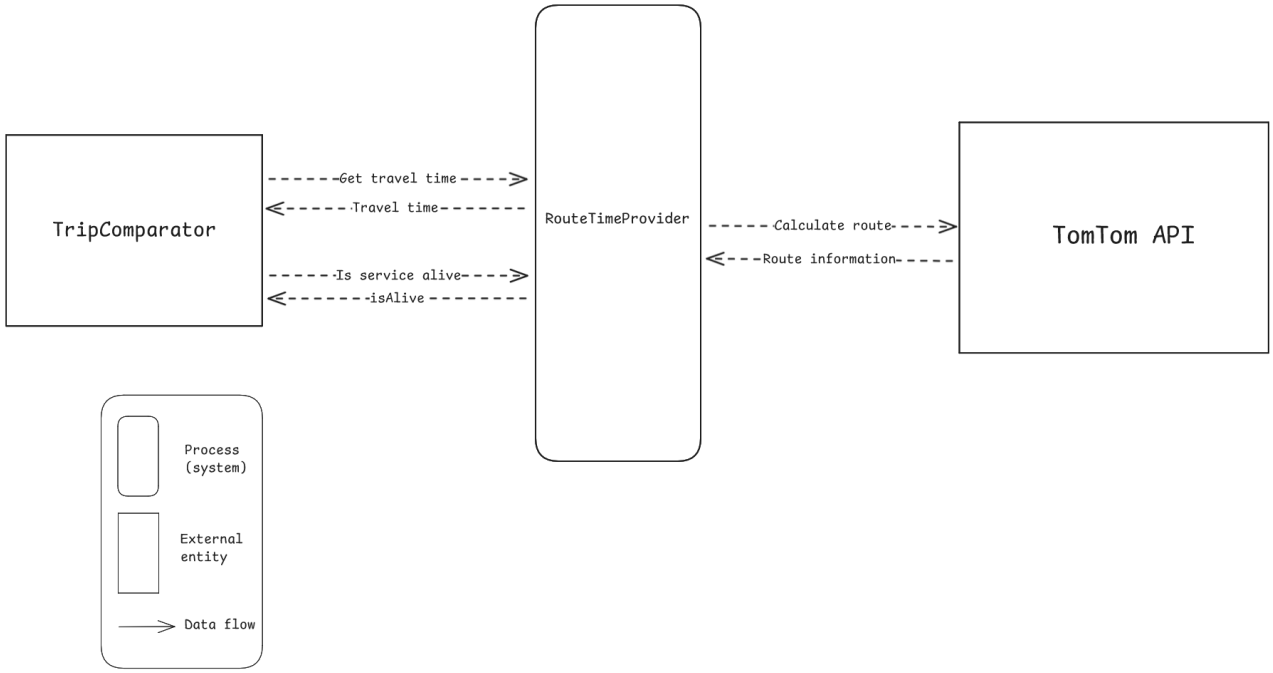
Diagramme de séquence /3

Diagrammes de contexte de haut niveau

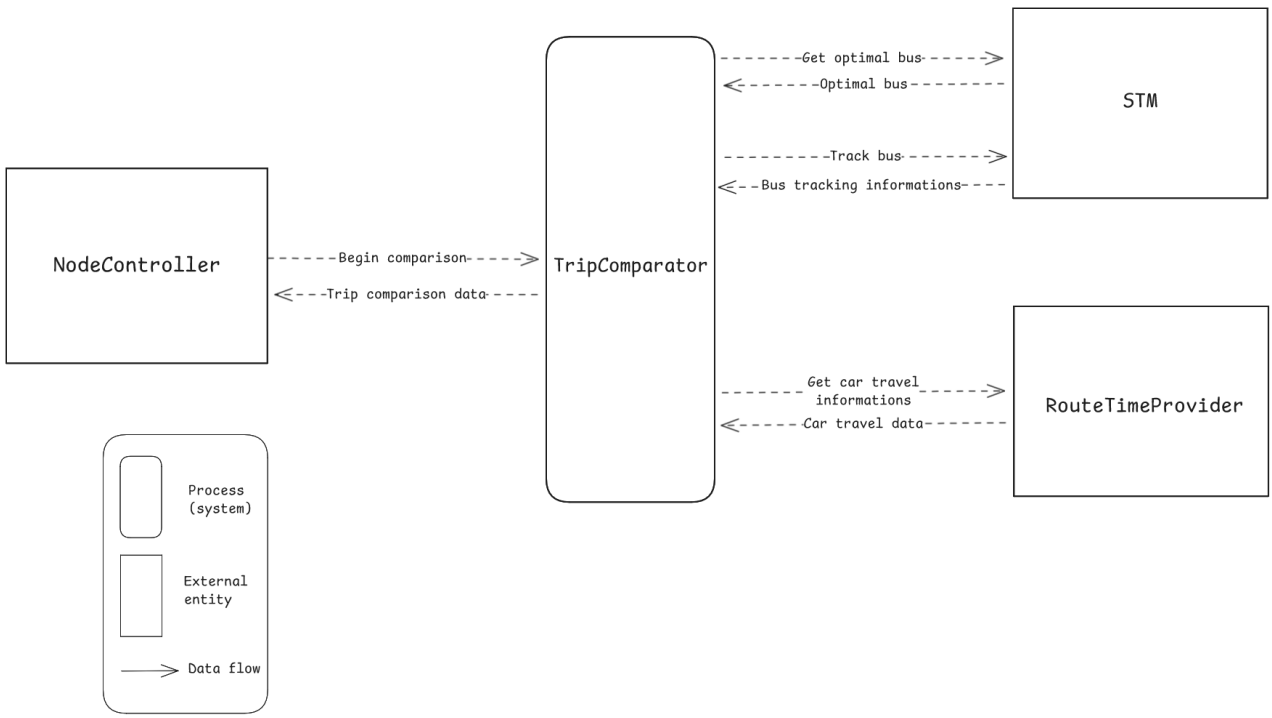
Microservice STM /3



Microservice RouteTimeProvider /3



Microservice TripComparator /3



Questions générales /12

1. L'ajout de la base de donnés est une amélioration au système. Quel attribut de qualité est amélioré par l'ajout d'une base de données? Choisissez un attribut de qualité vu en classe ou présent dans le livre du cours **Software Architecture in Practice**. Expliquez votre réponse. **/3**

> L'ajout d'une base donnée permet d'améliorer la performance du système. En effet, avant l'utilisation d'une base de donnée, le système devait mettre en mémoire vive une grande quantité d'information concernant les horaires des autobus, les trajets des autobus, la localisation des arrêt où les autobus arrête prendre des passagers, etc. De ce fait, le système utilisait plus de 8 gigaoctet de mémoire vive,

> rendant le programme impossible à exécuter sur une machine ayant moins de 16 gigaoctets de mémoire vive.

2. Lorsqu'une requête de comparaison du temps de trajet faite sur le Dashboard est envoyée au NodeController, différentes requêtes sont échangées sur le réseau. Dans la minute suivant l'envoi de la requête initiale, combien de requêtes sont échangées entre les microservices **NodeController**, **TripComparator**, **RouteTimeProvider** et **STM** et avec les API externes de la STM et de TomTom ? Comptez chaque requête échangée. Ainsi, si une requête est envoyée 10 fois, elle comptera comme 10 requêtes. Décrivez également rapidement les différentes requêtes échangées. **/4**

   2.1 Dans le code source du projet, nous pouvons ajuster le taux de lancement des requêtes envoyées par le microservice TripComparator vers le microservice STM. Expliquez comment nous pouvons nous y prendre pour ce faire en donnant un ou des extraits de code. /2

   > Dans le code source du service TripComparator, l'intervale des requêtes envoyées vers le service STM sont est dictée par un Periodic Timer. `private readonly PeriodicTimer _periodicTimer = new(TimeSpan.FromMilliseconds(50));` On peux remarquer qu'une requête sera faite à chaque 50 millisecondes. Il est donc possible d'ajuster le taux de lancement des requêtes envoyées en changeant le taux de sondage.

3. Proposez une tactique permettant d'améliorer la disponibilité de l'application lors d'une attaque des conteneurs de computation (**TripComparator**, **RouteTimeProvider**, et **STM**) lors du 2e laboratoire. **/3**

> Afin d'améliorer la disponibilité du système, l'utilisation des répliques de conteneurs avec un distributeur de charge serait une tactique possible. L'utilisation de réplique permet en effet de remplacer un services lorsque ce dernier n'est plus utilisable.

Conclusion du laboratoire /2

> En conclusion, le système est composé de plusieurs microservices qui communiquent entre eux afin de fournir une comparaison du temps que prend un trajet en voiture et en autobus. Pour y arriver, le système a besoin d'une grande quantité d'information qui doit être persistée dans une base de donnée afin de rendre l'application plus performante. De plus, il serait préférable d'utiliser la tactique des répliques de conteneurs en tandem avec un distributeur de charge afin d'améliorer la disponibilité du système.