Documentation d'Architecture Logicielle

Document d'architecture logicielle

LOG430-01 École de technologie supérieure
Documentation d'Architecture Logicielle (SAD)

LOG430-01 | Hiver 2022
Coordonateurs: Mathieu Massicotte, Olivier Brassard, Philippe Doyon, P-A Heredero, Simon Boyer

Table des matières

Overview

Feuille de route et vue d'ensemble de la documentation

Les sous-sections de cette page fournissent des informations qui aideront les lecteurs ou les utilisateurs du document d'architecture logicielle (SAD) à trouver rapidement les informations qui leur sont utiles.

Portée et résumé du document

Ce document décrit l'architecture du système MontrealLiveTraffic™. Cette application web va permettre de calculer le temps de trajet grâce au capteur de la ville de Montréal ou a des services externes. Pour le moment, le projet est limité a une section de la rue Notre-Dame.

Le projet a une architecture en microservice. Elle est inspiré d'une ébauche fournie par Yvan Ross dans le cadre du cours LOG-430.

Organisation du document d'architecture

Ce document est divisé en cinq sections principales :

1. Feuille de route et vue d'ensemble de la documentation : fournit des informations sur ce document et son public cible. Il fournit la feuille de route et un aperçu du document.
2. Contexte d'architecture : fournit des informations sur l'architecture du logiciel. Il décrit le contexte et la justification de l'architecture logicielle. Il explique les contraintes et les influences qui ont conduit à l'architecture actuelle et décrit les principales approches architecturales qui ont été utilisées dans l'architecture.
3. Vue et mappages entre les vues : fournit des informations sur les vues architecturales disponibles. Il décrit les vues architecturales disponibles et les mappages entre les vues, c'est à dire les relations existantes entre les différentes vues.
4. Documents de référence: fournissent des informations et des références aux documents qui sont cités ailleurs dans ce SAD.
5. Glossaire et acronymes: contient des définitions de termes techniques utilisés dans la documentation qui peuvent ne pas être familières à tous les lecteurs.

Comment une vue est documentée

1. **Présentation principale** : Les éléments et les relations de cette vue sont présentés sous forme de diagramme ou de représentation non graphique.

2. **Catalogue d'éléments** : Cette section contient un dictionnaire des éléments de la présentation principale. Elle fournit des informations et propriété supplémentaire pour facilité la compréhention.
3. **Diagramme de contexte** : Présentation sous forme de diagramme qui montre les limites du système.
4. **Guide de la modifiabilité** : Décrit les mécanismes de modifiabilité utilisé dans la partie du système montré dans la vue. Cela comprend aussi la manière et le moment où ces mécanismes peuvent être exercés.
5. **Raisonnement** : Justification de toute décision de conception importante dont la porté est limitée à cette vue. Les alternatives importantes sont aussi documentées.
6. **Vue associé** : Lien vers la vue parent et les vues enfants (si elles existent).

Context

Contexte Architectural

Contexte du problème

Les sous-sections suivantes expliquent les contraintes et les besoins qui ont influencé l'architecture du système.

Aperçu du système

Le système **MontrealLiveTraffic™** a comme objectif l'amélioration du calcul d'un temps de trajet entre deux points dans la ville de Montréal. En effet, dans le but de se positionner comme une des villes les plus intelligentes du monde, la ville de Montréal à mis à la disposition du public des données de capteurs en temps réel qui pourraient améliorer la qualité des estimation du temps requis pour effectuer un certain trajet. Dans le cadre de ce projet, le système se limitera dans un premier temps aux capteurs de température et radars situés sur la rue Notre-Dame.

Dans le but de tester différents algorithmes de calcul, le système permet de comparer des trajets provenant de différentes sources de données, que ce soit à partir des données de capteurs ou de services externes tels que Google Maps. Par ailleurs, le système permet de simuler les données des capteurs de la ville de Montréal afin de pouvoir tester les implémentations même si les capteurs ne sont pas disponibles.

L'architecure du système est composées de plusieurs micro-services qui permettent notamment d'accomplir les tâches suivantes :

- Calcul du temps de trajet entre deux points à l'aide des données de capteurs
- Calcul du temps de trajet entre deux points à l'aide de services externes (Google Maps, Bing Maps, Mapbox)
- Simulation des données des capteurs de la ville de Montréal
- Service de comparaison des temps de trajet entre deux points
- Interfaces utilisateurs

Contexte

Ce document a pour but de donner une vue d'ensemble des micro-services composant le système **MontrealLiveTraffic™**. L'objectif caché derrière le choix de décomposer les systèmes en micro-service est de développer un système modulable et fiable. En procédant de cette façon, nous pourrons réaliser plus facilement les nombreux attributs de qualités requises par le client. De plus, comme nous sommes plusieurs équipes à travailler sur le projet et que nous avons des expériences différentes, le fait de créer plusieurs micro-services plutôt qu'une application monolithique offre à chaque équipe de choisir la pile de technologies la plus appropriée, tout en profitant d'interfaces web communes afin d'interconnecter les micro-services pour former le système complet.

Note référez vous à la section Vue Globale du Système pour plus de détails sur les micro-services.

Exigences principales

Pour plus de détails sur les attributs de qualité et cas d'utilisation, voir la section Exigences du projet.

Principaux cas d'utilisation

- CU01. Comparer les temps de trajet
- CU02. Possibilité de mettre le chaos dans les services en
- CU04. Perturber le trafic à l'aide d'une interface usager pour simuler un accident ou une période de pointe sur les données de simulation
- CU05. Simulation de donnée GPS
- CU06. Simulation usager multiple
- CU07. Service externes
- Util: Traidement/filtrage des données de simulation de la ville de Montréal

Attribut de qualité

Disponibilité

- AQD1. Monitorage
- AQD2. Ping/écho
- AQD3. Heartbeat
- AQD4. Redondance passive
- ADQ5. Redondance active

Modifiabilité

- AQM1. Modifier sélection et intersection
- AQM2. Interface avec autre équipe
- AQM3. Généralisation redondance passive
- AQM4. Généralisation redondance active

Testabilité

- AQT1. Interface spécialisée
- AQT2. Abstraire, source de données.
- AQT3. Limiter opération non déterministe
- AQT4. Record/playback
- AQT5. Gestion centralisée des logs + id message

Convivialité

- AQU1. UI
- AQU2. Aggregation

Performance

- AQP1. Optimisation sur la latence
- AQP2. 1000 clients

Interopérabilité

- AQI1. Utilisation des microservices des autres équipes
- AQI2. Découverte dynamique de service

Sécurité

- AQS1. Authentification
- AQS2. Maintain audit trail

Contexte de la solution

Les sous-parties de cette section expliquent les raisons qui ont conduit à la conception de l'architecture telle qu'elle-est et fournissent une justification de l'adéquation de l'architecture aux objectifs de qualité qui lui sont fixés.

Approche architecturale

L'application sera divisée en différent services sur la base des fonctionnalités et des responsabilités. Une telle séparation réduit la dépendance entre les modules et nous permet les développer indépendamment. En outre, l'identification au préalable des interfaces et contrats de communication permet le développement des différents services par des équipes différentes. Les services du système sont documentés en détail à la section Vue Globale du Système.

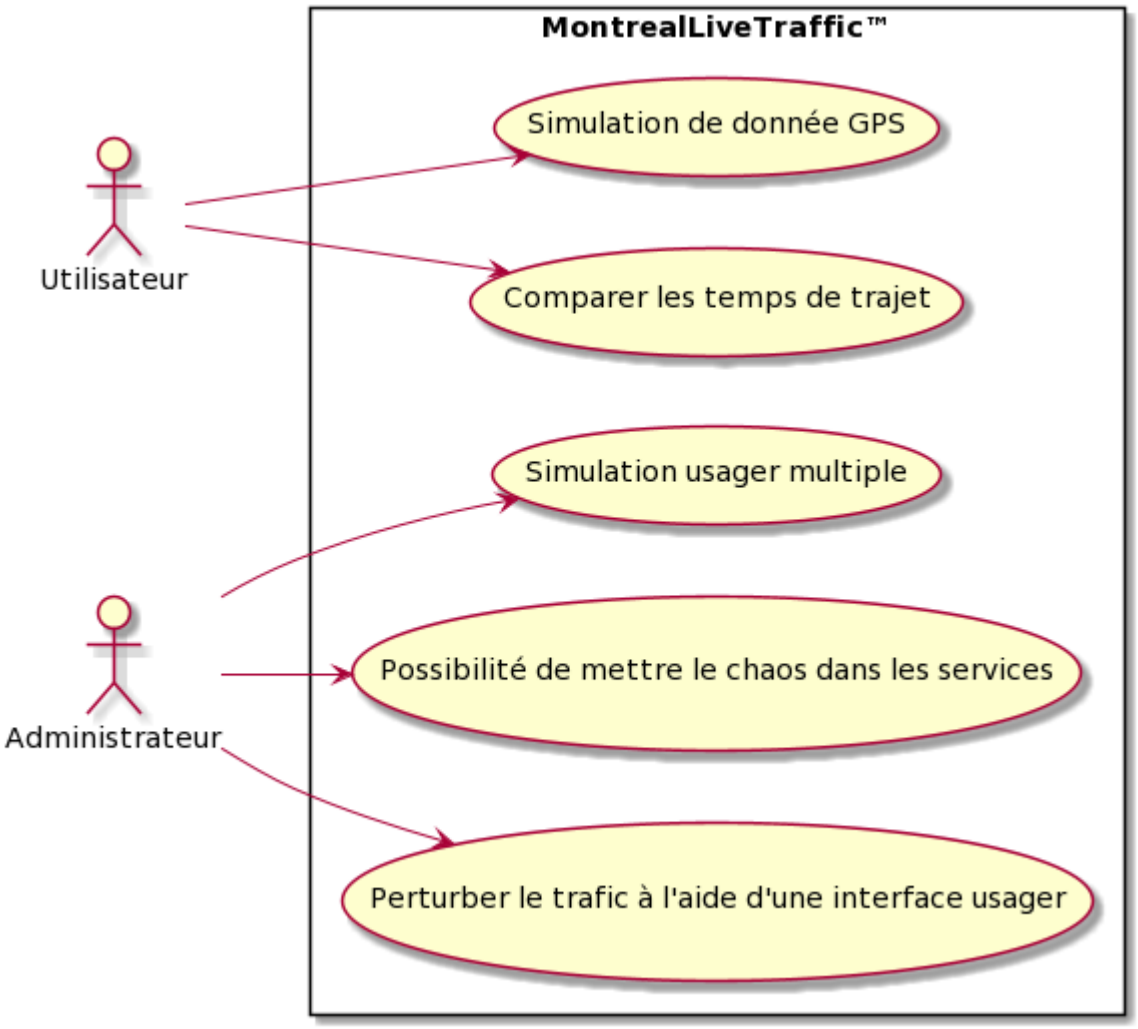
Par ailleurs cette séparation des responsabilités permet de prioriser le développement des fonctionnalités les plus essentielles et rend chaque module plus générique. Cela facilite la mise en place de tests unitaires et de tests d'intégration et nous permet de mieux répondre aux attribus de qualité comme la redondance active, la redondance passive et la disponibilité.

Nous utilsons des conteneurs Docker afin de faciliter le déploiement, la surveillance (monitoring) et la testabilité du système. Chaque service est donc isolé dans son propre conteneur, ce qui nous offre une grande flexibilité pour le développement et la mise en production. C'est aussi pour cette raison que chaque équipe est libre de choisir la stack de technologies la plus adaptée à leurs besoins.

Résultat des analyses

Il n'y a pas eu d'analyse, le client nous a imposé l'utilisation de l'architecture par micro-services.

Mise en correspondance des exigences et de l'architecture



Ce document se concentre sur les cas d'utilisation suivants. Sous chaque cas d'utilisation, vous trouverez les micro-service qui comblent lesbesoins associés à ceux-ci.

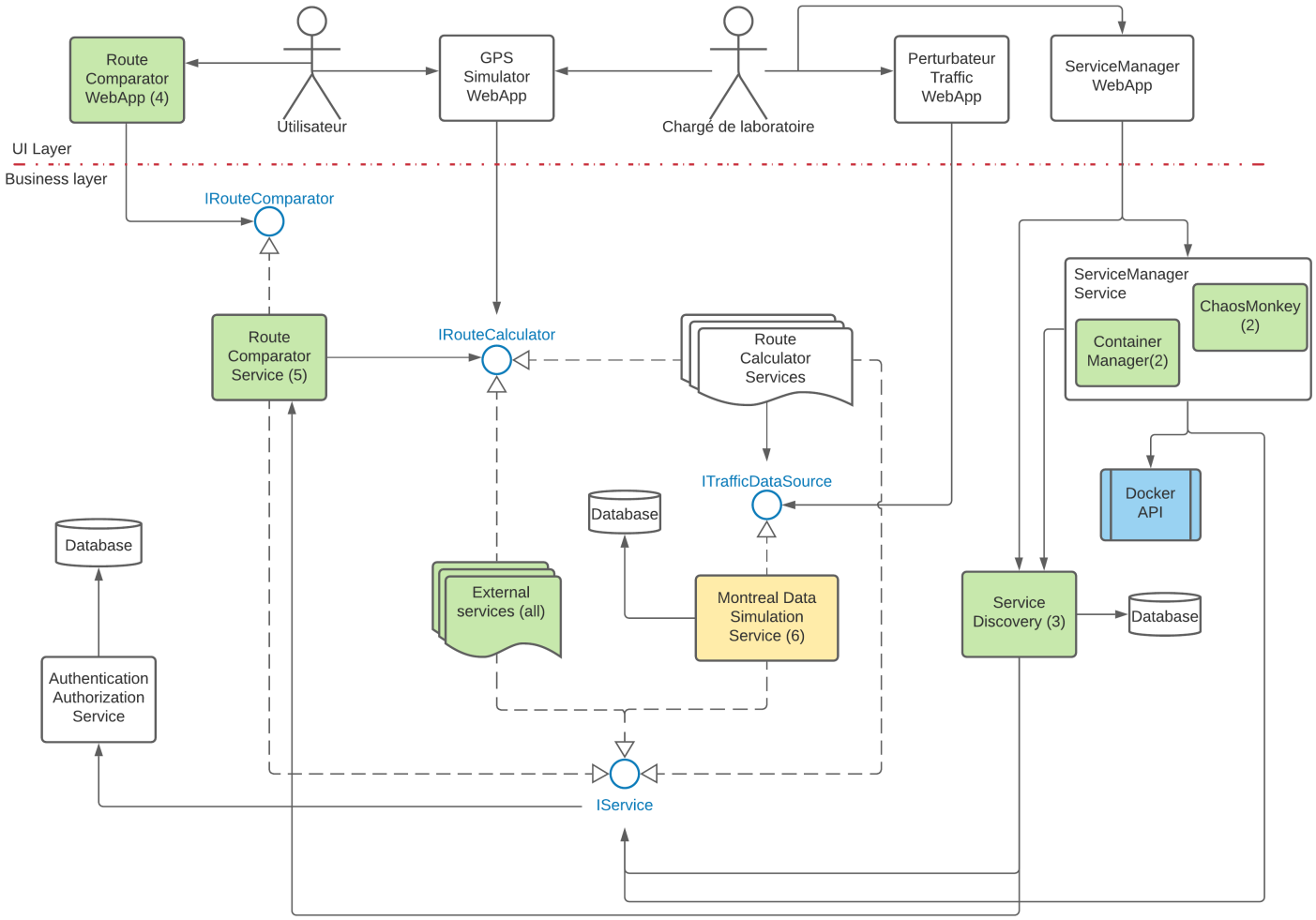
- CU01. Comparer les temps de trajet
  - RouteComparatorWebApp : Application web qui permet de visualiser dans un graphique les données des différents services.
  - RouteComparatorService : Fournit les données à l'application web.
  - RouteCalculatorService : Calcule le temps de trajet entre deux point sur la rue Notre-Dame à un moment donné.
  - ExternalServices : Calcule le temps de trajet entre deux point sur la rue Notre-Dame à un moment donné.
  - ServiceDiscovery : Permet l'interconnection entre les services.
  - SimulationDataService : Fournit les données au RouteCalculatorService.
- CU02. Possibilité de mettre le chaos dans les services
  - ChaosMonkey: Permet de tuer ou de changer la latence des micro-services du système.
  - ServiceDiscovery : Permet l'interconnection entre les services.

- CU04. Perturber le trafic à l'aide d'une interface usager
  - PertubateurTrafficWebApp : Permet de simuler un accident, une fermeture de voie...
- CU05. Simulation de donnée GPS
  - GpsSimulatorWebApp : Simule un GPS en déplacement, en affichant en "temps réel" le temps de trajet restant.
  - RouteCalculatorService : Calcule le temps de trajet entre deux point sur la rue Notre-Dame à un moment donné.
  - SimulationDataService : Fournit les données au RouteCalculatorService.
  - ServiceDiscovery : Permet l'interconnection entre les services.
- CU06. Simulation usager multiple
  - Tout les services sont mis en oeuvre afin de parvenir à cette tâche.
- CU07. Services externes
  - GoogleMapService: Retourne les données de trajet fournies par Google Map
  - BingMapService: Retourne les données de trajet fournies par Bing Map
  - MapBoxService: Retourne les données de trajet fournies par MapBox

Architecture-globale

Présentation primaire

La vue suivante montre l'ensemble des services qui seront déployés dans le cadre du projet. Chaque service correspond à un ou plusieurs conteneurs docker.



Les cases en vert sont des services complétés, les autres sont des services à compléter

Catalogue des éléments

[RouteComparatorWebApp]

Web UI qui affiche et compare les calculs de route d'une origine vers une destination selon des services externes et/ou des services MQTT qui implémentent IRouteComparator (sélectionner ce qu'il faut comparer).

GpsSimulatorWebApp

Web UI qui simule une application GPS selon un fournisseur IRouteCalculator sélectionné.

PertubateurTrafficWebApp

Élément perturbant le trafic pour voir l'impact sur le temps total de trajet. Intéragit avec un ITrafficDataSource pour initier ces événements.

ServiceManagerWebApp

Gestionaire de service et générateur de chaos qui peut modifier la latence ou détruire un microservice selon différents critères. Doit aussi pouvoir détruire/faire arrêter/planter/terminer le processus de ServiceDiscovery pour que celui soit temporairement non disponible.

ServiceManager

Gestionaire de service qui contrôle les services déployés et offre un service de redondance passive pour les autres services.

Contient aussi l'API du générateur de chaos décrit dans ServiceManagerWebApp.

RouteComparatorService

Comparateur des calculs du temps de trajet entre deux coordonnées GPS. Le calcul se fait sur le tronçon de la rue Notre-Dame seulement. On lui fourni une liste de services implémentant IRouteCalculator à comparer et RouteComparatorService fera les requêtes nécessaire pour comparer les reésultats et retourner les informations de comparaison (à préciser).

RouteCalculatorService

À implémenter par chaque équipe

Calculateur du temps de trajet à partir des données MQTT. Elles seront fournies à travers l'interface ITrafficDataSource. L'algorithme est à la discretion de chaque équipe.

ExternalServices

Service de calcul de trajet externe. Ce service communiquera avec le service externe et retournera les informations selon le format de l'interface IRouteCalculator. Chaque service sera implémenté à 2 reprises selon la disposition suivante:

Fournisseur	Nom du service	Équipes
Google Maps	GoogleMapsService	2, 3
Bing Maps	BingMapsService	4, 5

Fournisseur	Nom du service	Équipes
Mapbox	MapBoxService	6

Montreal Data Simulation Service

Service permettant d'analyser les fichiers historiques MQTT de la ville de montréal, d'enregistrer ces informations dans une base der données puis de servir ces données selon des requêtes à travers l'interface ITrafficDataSource. On précisera une intervalle temporelle lors de ladite requête afin d'identifier quelles données retourner.

ServiceDiscovery

Service recevant des appels d'enregistrement de la part d'un IService. Ensuite vérifie l'état (health checks) de ces services de façon régulière. Permet ensuite aux autres services d'utiliser le ServiceDiscovery afin d'avoir une répertoire des services actifs et en bonne santé.

Interfaces concernées

Liste des interfaces concernées par la vue.

- A0.1 IService
- A0.2 IRouteCalculator
- A0.3 ITrafficDataSource
- A0.4 ServiceDiscovery

Diagramme de contexte

Cette vue représentant le système de très haut niveau, la vue principale a ici aussi le rôle de diagramme de contexte.

Guide de variabilité

Les configurations possibles de chacun des services sont décrites dans chacunes de leurs vues.

Justification

La séparation en plusieurs petits modules permet de respecter la loi de Conway et de mieux distribuer la charge de travail entre les équipes. De plus, séparer chaque tâche ainsi que de séparer le UI de la logique applicative permet une meilleure séparation des responsabilités (separation of concerns).

Vues et Mappage entre les vues

Vues et Mappage entre les vues

Vues et spécifications remplies

Vues équipe 2

Vue	Cas d'utilisation	Attributs de qualité
IServiceBaseImplementationView	CU02	AQT1
Service Manager		AQM2
ChaosMonkey	CU02	
Redondance Passive		AQM3,AQD4
GMaps Service	CU07	

Vues équipe 3

Vue	Cas d'utilisation	Attributs de qualité
GoogleMapsService	CU07	AQD2, AQD3, AQT1
ServiceDiscovery	CU02, CU01	AQD1, AQD2, AQD3, AQI2, AQT1

Vues équipe 4

Vue	Cas d'utilisation	Attributs de qualité
Comparaison de trajet avec des services externes	CU01 & CU07	
BingMaps Service Externe	CU07	
RouteComparator WebApp	CU01	

Vues équipe 5

Vue	Cas d'utilisation	Attributs de qualité
Interface spécialisée		AQT1
Limiter les opérations non déterministes		AQT3
BingMaps	CU07	
RouteComparatorService	CU01	

Vues équipe 6

Vue	Cas d'utilisation	Attributs de qualité
MapBoxService	CU07	AQT1
MontrealSimulationDataService	Util	AQT1

Vues globales

Vue	Cas d'utilisation	Attributs de qualité
Architecture Globale	N/A	N/A

Vues et Vues

Équipe 2

Vue 1	Vue 2	Lien
Service Manager	ChaosMonkey	Parent-Enfant
Service Manager	Redondance Passive	Parent-Enfant

Équipe 4

Vue 1	Vue 2	Lien
Comparaison de trajet avec des services externes	BingMaps Service Externe	Parent-Enfant

Vue 1	Vue 2	Lien
Comparaison de trajet avec des services externes	RouteComparator WebApp	Parent-Enfant

Vues et interfaces

Vues équipe 2

Vue	Interfaces
IServiceBaseImplementationView	IService (coordonnateurs)
GMapsExternalService	IRouteCalculator (coordonnateurs)
ChaosMonkey	ChaosmonkeyApi (équipe 2)
ServiceManager	IService (coordonnateurs)

Vues équipe 3

Vue	Interfaces implémentées
GoogleMapsService	IService, IRouteCalculator
ServiceDiscovery	IService, IServiceDiscovery

Vues équipe 4

Vue	Interfaces
Comparaison de trajet avec des services externes	
BingMaps Service Externe	IService (coordonnateurs),IRouteCalculator (coordonnateurs)
RouteComparator WebApp	IRouteComparator (équipe 5)

Vues équipe 5

Vue	Interfaces

Vue	Interfaces
Interface spécialisée	BingMaps (équipe 5)
Limiter les opérations non déterministes	RouteComparatorService (équipe 5)
RouteComparatorService	IService (coordonnateurs)
BingMaps	IService (coordonnateurs)

Vues équipe 6

Vue	Interfaces
MapBoxService	IService (coordonnateurs), IRouteCalculator (coordonnateurs)
MontrealSimulationDataService	IService (coordonnateurs), ITrafficDataSource (équipe 6)

API Reference

LOG430

API Version: 1.0.0

Documentation de base pour les micro services du laboratoire de LOG430.
École de Technologie Supérieure - Hiver 2022.

En cas de question : @Coordonnateur sur discord

INDEX

API

1. IROUTECALCULATOR

Service that allow comparison of routes

1.1 GET /route/{origin}/{destination}

Get travel time between to points

Return the estimated route time between two points for a given provider.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*origin	string	Coordinates of the origin point (Format : LAT,LONG)
*destination	string	Coordinates of the destination point (Format : LAT,LONG)

RESPONSE

STATUS CODE - 200: Information about the time required to drive from origin to destination

RESPONSE MODEL - application/json

```
{
  date         string
  origin       string
  destination  string
  duration     number  Duration in minutes
  distance     number  Distance in meters
}
```

2. ISERVICE

Basic endpoints that all micro services should implements

2.1 GET /api/info

Get API info

Retourne les informations de l'API

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
```

Api info response which allows to check that the service is still active

```
  serviceName    string  The name of the micro service
  teamNumber     number  Number of the team that developed the service
  currentLatency string  Current service latency
}
```

2.2 PUT /api/latency

Set API latency

Permet de mettre à jours la latence pour effectuer des tests.

REQUEST

REQUEST BODY - application/json

```
{
  latency* number  Permet de modifier la latence (en miliseconde). Pour réinitialiser la latence, mettre la valeur '0'
}
```

RESPONSE

STATUS CODE - 200: OK

STATUS CODE - 400: BAD REQUEST

RESPONSE MODEL - application/json

```
{
  statusCode number
  message    string
}
```

Documents de référence

Documents de référence

Ce document est basé sur un modèle mis à disposition par l'École de science informatique de l'Université de Carnegie Mellon.
https://wiki.sei.cmu.edu/confluence/display/SAD/Main+Page

02 février 2022

Matériel de référence

Auteur	Référence
Bachmann 2010	Garlan, Bachmann, Bass, Ivers, Little, Stafford, Clements, Merson, Nord, Documenting Software Architectures: Views and Beyond, 2nd_Edition, Addison-Wesley, 2010

Glossaire et acronymes

Glossaire et acronymes

UI : Interface utilisateur