MedHead

HYPOTHESE DE VALIDATION DE PRINCIPE

MedHead

HYPOTHESE DE VALIDATION DE PRINCIPE

---

1. Introduction

Ce document permet d'évaluer la conformité avec la vision de l'architecture. Il décrit un processus qui sert à démontrer que le système de gestion d'urgence des patients de MedHead pour lequel la POC a été réalisé répond répond bien aux exigences de l'architecture cible établie.

2. Déclaration d'hypothèse

Ce travail de réalisation de POC pour le sous-système d'intervention d'urgence en temps réel permet d'améliorer la qualité des traitements d'urgence et de sauver plus de vies et de gagner la confiance des utilisateurs.

Sa réussite passe par la validation d'un certains nombre de principes d'architecture décrits dans le document «principeArchitecture», mais aussi par les résultats concrets suivants :

a. lus de 90 % des cas d'urgence sont acheminés vers l'hôpital compétent le plus proche du réseau ;

b. Le temps moyen de traitement d'une urgence passe de 18,25 minutes (valeur actuelle) à 12,00 minutes (valeur souhaitée)

c. L'obtention d'un temps de réponse de moins de 200 millisecondes avec une charge de travail allant jusqu'à 800 requêtes par seconde, par instance de service ;

d. La mise en œuvre explique les normes qu'elle respecte et pourquoi ;

e. Les instructions pour mettre en production la PoC sont fournies ;

f. La mise en œuvre est terminée dans le délai imparti.

3. Exigences convenues de la POC

Les exigences suivantes ont été convenues lors de la définition de cette hypothèse :

a) Fournir une API RESTful qui tient les intervenants médicaux informés en temps réel sur : le lieu où se rendre et ce qu'ils doivent faire.

b) S'assurer que toutes les données du patient sont correctement protégées.

c) S'assurer que votre PoC est entièrement validée avec des tests d'automatisation reflétant la pyramide de test (tests unitaires, d'intégration, d'acceptation et E2E) et avec des tests de stress pour garantir la continuité de l'activité en cas de pic d'utilisation.

d) S'assure que la PoC peut être facilement intégrée dans le développement futur : rendre le code facilement partageable, fournir des pipelines d'intégration et de livraison continue (CI/CD) et documenter votre stratégie de test.

e) S'assurer que les équipes de développement chargées de cette PoC sont en mesure de l'utiliser comme un jeu de modules de construction pour d'autres modules.

4. Description de l'architecture de la POC

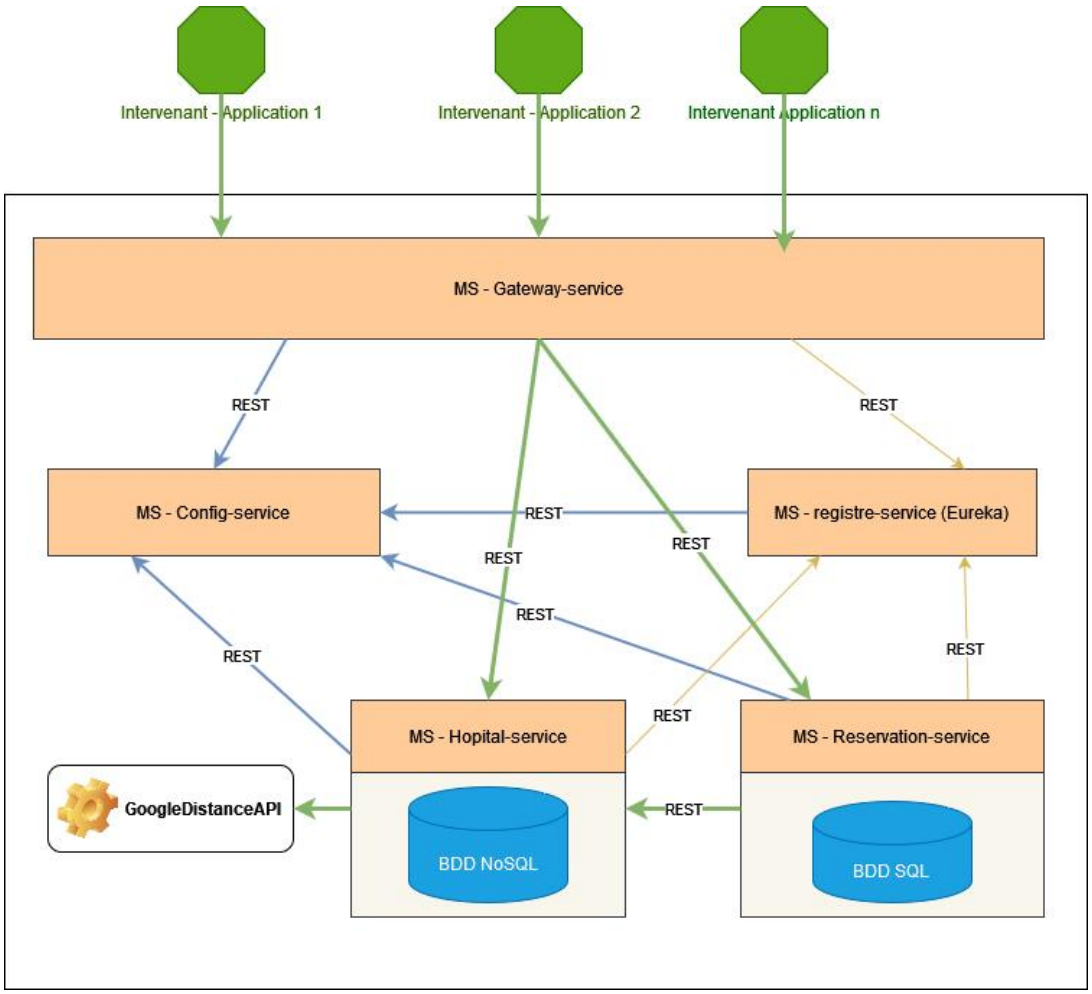

Image 1 - architecture en microservices - POC

Description des composants

Le tableau ci-dessous nous présente la liste des composants à ajouter pour optimiser l'architecture actuelle de l'entreprise Foosus

Composants	Description
MS Gateway-seervice	(Également appelée passerelle API) est le point d'entrée unique pour les API et microservices backend définis (qui peuvent être à la fois internes et externes). Assise devant les API, l'API Gateway agit en tant que protecteur, renforçant la sécurité et assurant l'évolutivité et la haute disponibilité.
MS Registre-service	Répartition de charge, est une technologie conçue pour distribuer la charge de travail entre différents serveurs ou applications. Le but : optimiser la performance globale de l'infrastructure, son rendement et sa capacité.
MS Config-service	Ce composant centralise la configuration de tous nos microservices
MS Hopital-service	Ce composant permet de rechercher l'hopital le plus proche du lieu d'incident en fonction de la spécialité demandée.
MS Reservation-service	Ce composant permet aux intervenants de réserver un hôpital
Service facturation	Ce composant permet de gérer la facturation des fournisseurs.
Google Distance API	Ce composant permet de calculer la distance entre le lieu d'incident et les adresses des hôpitaux présent dans la base.
API REST	Une API REST (également appelée API RESTful) est une interface de programmation d'application (API ou API web) qui respecte les contraintes du style d'architecture REST et permet d'interagir avec les services web RESTful.

5. Evaluation de la conformité

Pour pouvoir évaluer et valider l'hypothèse, nous avons défini l'ensemble de checklists suivants:

1. Services tiers logiciels et midleware checklist.
2. Application checklist.
3. Gestion des données checklist.
4. Sécurité checklist.
5. Infrastructure checklist.
6. Méthodes et outils checklist.

Services tiers logiciels et middleware checklist

Questions d'évaluation	Méthodode	Résultats
Est-ce qu'il y a un seul point d'entrée de microservices?	Utilisation d'un gateway	Oui
Est-ce que les protocoles de communication entre les microservices sont indépendants des plate-formes?	Utilisation de Http/ Https	Oui
Est-ce qu'un API externe est utilisé pour calculer la distance entre le lieu d'incident et les hôpitaux compétent	Présence de GoogleMatrixApi	Oui

Applications checklist

Questions d'évaluation	Méthodologie	Résultats
Est-ce que la couverture des tests est acceptable?	Utilisation de Jacoco	Oui (plus de 80%)
Est-ce que les tests sont automatisés	1. JUnit 2. JMeter 3. Jenkins	Oui
Est-ce les différents types de tests sont validés	1. Tests unitaires 2. Tests intégrations 3. Tests acceptation 4. Tests montée en charge	Oui
Est-ce que l'application est scalable?	Utilisation d'une architecture microservice	Oui
Est-ce qu'on peut utiliser comme un jeu de modules de construction pour d'autres modules		Oui

Gestion des données checklist

Questions d'évaluation	Méthodologie	Résultats
Est-ce que la recherche d'un hôpital compétent est simple et rapide?	Utilisation d'une base de donnée NoSQL	Oui
Est-ce que les données sont sécurisée?		
Est-ce que le format des données est conforme?		
Y-a-til une procédure de restauration des données en cas de perte?		

Sécurité checklist

Questions d'évaluation	Méthodologie	Résultats
Est-ce que les droits d'accès sont respectés?		
Est-ce que les données confidentielles et sensibles sont cryptées?		
Est-ce que l'accès aux microservices se passe par un unique point d'entrée	Utilisation d'un gateway	Oui
Est-ce que l'accès aux mircoservices sont sécurisés	Utilisation d'un Basic Auth où chaque requête doit s'authentifier grâce à un «username» et «password»	Oui

Infrastructures checklist

Questions d'évaluation	Méthodologie	Résultats
Est-ce que Docker est utilisé pour la conteneurisation ?		
Est-ce que la solution est déployée dans le cloud?		
Est-ce que les technologies utilisées sont conformes au standard de l'entreprise?	Socle technique : 1. Java 2. Spring cloud framework 3. Maven 4. SQL et NoSQL 5. Jenkins	Oui
Est-ce qu'il y a la répartition des charges		

Méthodes et outils checklist

Questions d'évaluation	Méthodologie	Résultat
Est-ce qu'il y a une procédure de mise en production de l'application?	Fichier ReadMe.md présent dans chaque microservice	Oui
Est-ce qu'il y a une procédure de déploiement de nouvelles fonctionnalités en production en cas de changement?		Oui