# Hypothèse de validation de principe

Client : Consortium MedHead

Projet : Preuve de concept (l'allocation de lits d'hôpital pour les urgences)

# Table des matières

# Contexte

Ce document présente l'hypothèse de validation de principe du POC pour le sous-système d'intervention d'urgence en temps réel.

# Déclaration d'hypothèse

Nous pensons que la mise en œuvre d'une preuve de concept pour le sous-système d'intervention d'urgence en temps réel par l'équipe d'architecture métier du Consortium

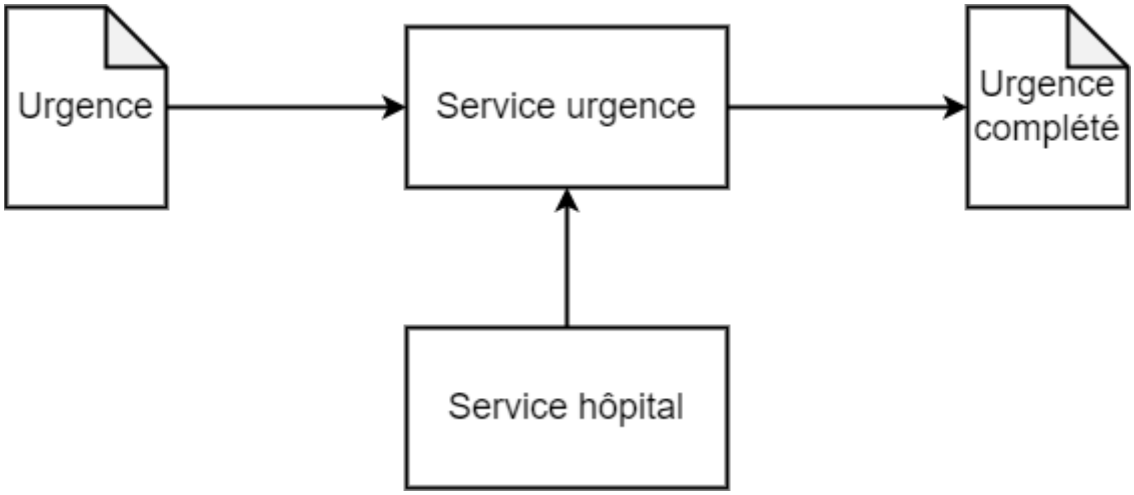
MedHead permettra :

- d'améliorer la qualité des traitements d'urgence et de sauver plus de vies ;
- de gagner la confiance des utilisateurs quant à la simplicité d'un tel système.

Nous saurons que nous avons réussi quand nous verrons que :

- ~~plus de 90 % des cas d'urgence sont acheminés vers l'hôpital compétent le plus proche du réseau ;~~
- ~~le temps moyen de traitement d'une urgence passe de 18,25 minutes (valeur actuelle) à 12,00 minutes (valeur souhaitée) ;~~
- ~~nous obtenons 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 ;~~
- la mise en œuvre explique les normes qu'elle respecte et pourquoi ;
- les instructions pour mettre en production la PoC sont fournies ;
- la mise en œuvre est terminée dans le délai imparti

# Comportement du systeme




Le principe de ce POC repose sur le schéma ci dessus :

1. Un systeme tiers envoie une requête d'urgence au service principal contenant :
   - Un identifiant publique du patient
   - La localisation du patient à aller chercher
   - La patologie dont souffre le patient
2. Le service d'urgence à pour fonction de compléter l'urgence en y ajoutant :
   - La localisation de hôpital le plus proche pouvant traiter la pathologie patient et aillant un lit disponibles
3. Le service hopital fournie sur demande les informations de l'hopital le plus proche aillant des lits disponibles en fonction de la pathologie à traiter et de la localisation d'origine de l'urgence
4. Ainsi, un urgence complété est publier en base en attendant des traitements subsidiaire comme la reservation de lit d'hopitaux pour le patient à traiter en urgence.

# Retour sur les exigences du POC

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

| No | Exigence | Retour d'expérience |
|---|---|---|
| 1 | 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. | **Se réalise au travers d'une requête demandant le traitement d'une urgence.** |

| No | Exigence | Retour d'expérience |
|---|---|---|
| 2 | S'assurer que toutes les données du patient sont correctement protégées. | **Le service d'urgence traite des identifiants public (clé public) de patient.** |
| 3 | 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. | **A l'heure actuelle, les tests sont majoritairements automatisé.** |
| 4 | S'assure que la PoC peut être facilement intégrée dans le développement fut rendre le code facilement partageable, fournir des pipelines d'intégration et de livraison continue (CI/CD) et documenter votre stratégie de test. | **Le repertoire de code fournie un fichier permettant de construire un pipeline CI/CD automatiquement.** |
| 5 | 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. | **Le repertoire de code est constitué de plusieurs modules réutilisables.** |

# Méthodologies

## Developpement

Trois modules ont été développé dans le cadre du POC. Ils sont composés : d'un service d'urgence, d'un service hopitaux ainsi que d'une api gateway. Le gateway a pour rôle de faire la correspondance entre les serveurs, aussi l'appel au service principal d'urgence se fait au travers de l'api gateway.

Les trois services sont développé à l'aide de Java Spring. Spring propose les services suivants (liste non-exhaustive) :

1. Découplage des composants. Moins d'interdépendances entre les différents modules.
2. Rendre plus aisés les tests des applications complexes c'est-à-dire des applications multicouches.
3. Diminuer la quantité de code par l'intégration de frameworks tiers directement dans Spring.
4. Permettre de mettre en oeuvre facilement la programmation orientée aspect.
5. Un système de transactions au niveau métier qui permet par exemple de faire du "two-phases-commit".

6. Un mécanisme de sécurité.
7. Pas de dépendances dans le code à l'api Spring lors l'utilisation de l'injection. Ce qui permet de remplacer une couche sans impacter les autres.
8. Une implémentation du design pattern MVC
9. Un support du protocole RMI. Tant au niveau serveur qu'au niveau du client.
10. Déployer et consommer des web-services très facilement.
11. Echanger des objets par le protocole http

## Tests

La méthodologie de test de pour le POC est composé de tests d'integration continue et E2E (*end-to-end*).

L'intégration continue repose souvent sur la mise en place d'une brique logicielle permettant l'automatisation de tâches : compilation, tests unitaires et fonctionnels, validation produit, tests de performances… À chaque changement du code, cette brique logicielle va exécuter un ensemble de tâches et produire un ensemble de résultats, que le développeur peut par la suite consulter. Cette intégration permet ainsi de ne pas oublier d'éléments lors de la mise en production et donc ainsi améliorer la qualité du produit. Ces tests sont fourni dans les couches de tests associés au service et l'aspect intégration est exécuté dans un pipeline CI/CD.

Les tests de bout en bout sont une technique utilisée pour vérifier si une application (site Internet, application mobile…) se comporte comme prévu du début à la fin. Le testeur doit se mettre dans la peau d'un utilisateur et dérouler les tests comme s'il utilisait véritablement l'outil mis à sa disposition. Cette technique permet de valider le fonctionnement du front. Mais aussi de vérifier son intégration avec le back-office et autres webservices. Pour le moment, ils sont réalisé à l'aide du outil externe Postman qui envoie des requetes des requetes Rest aux microservices d'urgence.

## Approche CI/CD

Comme vu dans la section des méthodologies de test, cette approche est la fusion des tests unitaire et d'integration. L'approche CI/CD permet d'augmenter la fréquence de distribution des applications grâce à l'introduction de l'automatisation au niveau des étapes de développement des applications. Les principaux concepts liés à l'approche CI/CD sont l'intégration continue, la distribution continue et le déploiement continu. L'approche CI/CD représente une solution aux problèmes posés par l'intégration de nouveaux segments de code pour les équipes de développement et d'exploitation (ce qu'on appelle en anglais « integration hell », ou l'enfer de l'intégration).

Plus précisément, l'approche CI/CD garantit une automatisation et une surveillance continues tout au long du cycle de vie des applications, des phases d'intégration et de test jusqu'à la distribution

et au déploiement. Ensemble, ces pratiques sont souvent désignées par l'expression « pipeline CI/CD » et elles reposent sur une collaboration agile entre les équipes de développement et d'exploitation.

L'outil utilisé pour réaliser le pipeline est Jenkins. Pour le moment, il exécute les builds de microservices et les tests d'integration. Plus tard, il pourra contenir les thématiques de qualité de code, la contenerisation et les test E2E.

# Résultats

- Le POC permet de simulé le comportement du service d'urgence et mets en lumière la nécessité de mise en œuvre d'un SIH (système d'information des hopitaux) performant.
- Le module hopitaux propose quelques notions utiles pour le futur SIH.
- Le SIH devra implémenté des fonctions recherches de trajets le plus en fonction des modes de transports : route et aérien.
- Test sur l'infrastructure cible permettront de validé les temps d'exécution.
- Des tests de monté en charge doivent également être fournie pour répondre au exicenge du point précédent.