

Hypothèse de développement d'une preuve de concept 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 Med Head 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 ;
- Que le temps moyen de traitement d'une urgence passe de 18,25 minutes (valeur actuelle) à 12,00 minutes (valeur souhaitée) ;
- Que 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 ;
- Que la mise en œuvre explique les normes qu'elle respecte et pourquoi ;
- Que les instructions pour mettre en production la PoC sont fournies ;
- Que la mise en œuvre est terminée dans le délai imparti.

Exemple de comportement et description de la capacité

Le sous-système d'intervention d'urgence en temps réel est destiné à recevoir une ou plusieurs spécialités médicales (voir les [Données de référence sur les spécialités](#)) et une banque de données d'informations récentes sur les hôpitaux afin de suggérer l'hôpital le plus proche offrant un lit disponible, associé à une ou plusieurs spécialisations correspondantes. Le lieu de l'incident d'urgence doit également être fourni.

Par exemple, SUPPOSONS trois hôpitaux, comme suit :

Hôpital	Lits disponibles	Spécialisations
Hôpital Fred Brooks	2	Cardiologie, Immunologie
Hôpital Julia Crusher	0	Cardiologie
Hôpital Beverly Bashir	5	Immunologie, neuropathologie diagnostique

ET un patient nécessitant des soins en cardiologie.

QUAND vous demandez des soins en cardiologie ET que l'urgence est localisée près de l'hôpital Fred Brooks

ALORS l'hôpital Fred Brooks devrait être proposé

ET un événement devrait être publié pour réserver un lit.

Exigences convenues de la PoC

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

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

Méthodologie

La documentation et la PoC qui en résulteront seront présentées aux membres du Conseil d'administration pour décrire les enseignements tirés de la PoC. Des rapports sur les méthodes CI/ CD seront présentés au personnel technique afin d'expliquer comment mettre à jour le système.

Afin de répondre aux exigences du POC, il faudra mettre en place des tests unitaires, d'intégrations et fonctionnels en accord avec le plan de test :

- 1- Les tests unitaires permettront de tester de petites unités de code permettant de valider leurs logiques.
- 2- Les tests d'intégrations auront tendance à tester les services en eux même et de rendre compte d'un fonctionnement du PoC pratiquement dans son entièreté.
- 3- Les tests fonctionnels permettront de tester le PoC dans son ensemble et de voir si celui-ci répond aux exigences voulues.

Un pipeline d'intégration continue s'assurant de la qualité du code :










- Le lancement automatisé des tests unitaires
- La génération du jar de l'API

Résultats obtenus

Tests unitaires et d'intégrations

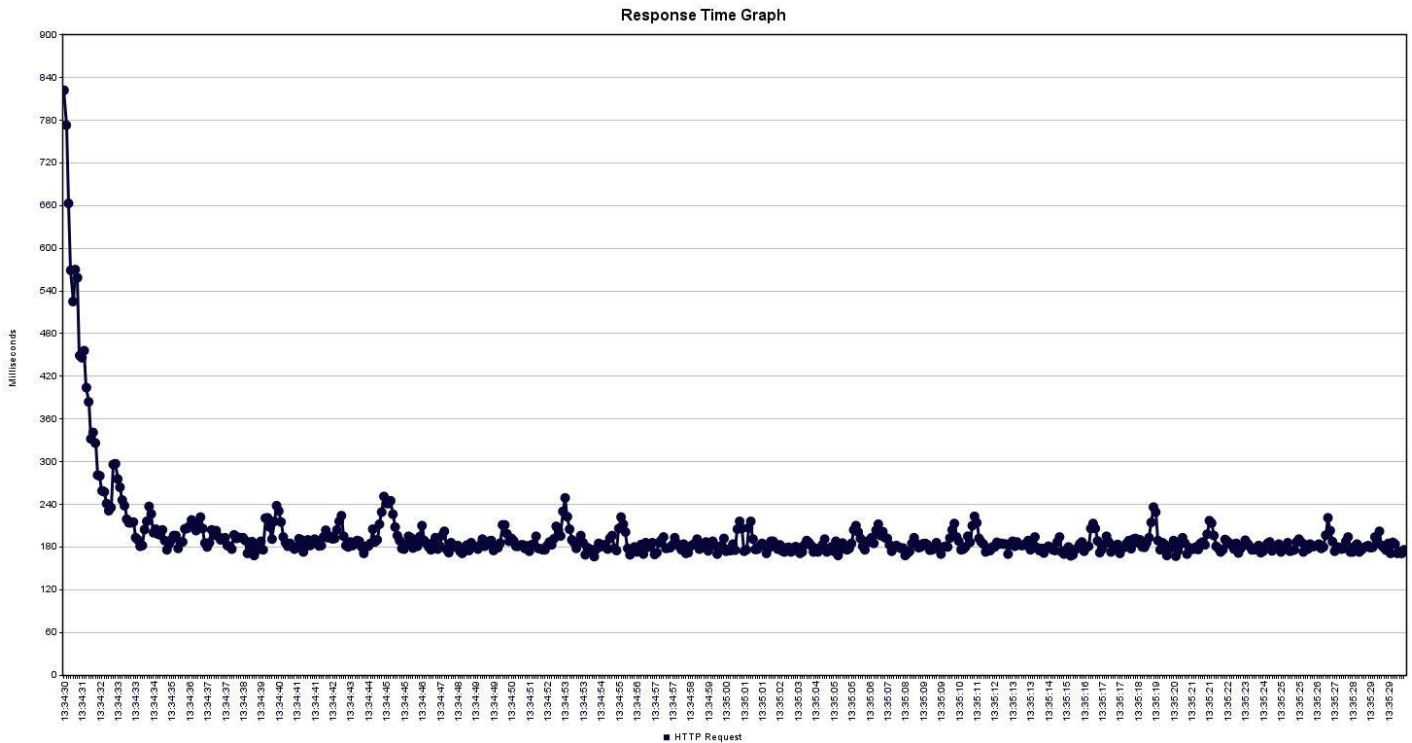
Les tests unitaires et d'intégrations présents sur la PoC sont exécutables via Maven et le résultat peut être obtenu via le plugin Surefire et JaCoCo. Ces derniers permettent de couvrir plus de 75% du code (capture d'écran ci-dessous).

urgencyManagement

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.medhead.urgencyManagement.configuration		38 %		12 %	5	10	23	38	1	6	0	2
com.medhead.urgencyManagement.repository		16 %		n/a	1	2	6	8	1	2	0	1
com.medhead.urgencyManagement		37 %		n/a	1	2	2	3	1	2	0	1
com.medhead.urgencyManagement.service		100 %		100 %	0	13	0	47	0	10	0	3
com.medhead.urgencyManagement.entity		100 %		n/a	0	32	0	60	0	32	0	2
com.medhead.urgencyManagement.controller		100 %		n/a	0	2	0	2	0	2	0	1
com.medhead.urgencyManagement.utils		100 %		n/a	0	2	0	2	0	2	0	1
Total	140 of 596	76 %	7 of 14	50 %	7	63	31	160	3	56	0	11

Le plus important reste la couverture du code concernant les controller et les services (possibles grâce aux tests d'intégrations et unitaires). En effet ce sont des tests permettant de s'assurer que la PoC dans son ensemble fonctionne comme elle le devrait. C'est pour cela que 100% du code est couvert et testé à ce niveau-là. Le code non couvert (configuration et repository) n'a pas été jugé aussi important à tester car ne possède pas de logique métier à proprement parlé.

Tests de stress



Les tests de stress sont concluants concernant les exigences de la PoC car l'implémentation actuelle ne permet pratiquement de répondre en dessous des 200 ms pour 800 utilisateurs sur une minute. En effet comme le démontre le graphique ci-dessus les premières requêtes mettent pratiquement 900 ms à répondre au système. Il semblerait qu'ensuite la chute observée du temps de réponse soit dû à un système de cache. Dans un second temps la réponse se stabilise aux alentours des 200 ms mais certaines sont malgré tout au-dessus. On estime que vu les conditions dans lesquelles a été effectué la PoC il est probable que des conditions de production permettent de répondre totalement aux tests de charge.

Conclusion

Les conclusions qui peuvent être tirées de cette PoC sont que :

- Fonctionnellement les exigences sont respectées. C'est-à-dire qu'une requête utilisateur renvoie l'hôpital le plus proche prenant en charge la pathologie du patient.
- Le système est protégé grâce à la mise en place de token.
- Le plan de tests est respecté et une partie de ce plan de tests est automatisé sur le pipeline.
- Les tests de stress sont concluants (réponse au-delà des 200 ms pour certaines requêtes) mais les performances peuvent être améliorées.

Solutions pour gérer la charge utilisateur :

1 - Proposer un cache local pour sauvegarder la liste des hôpitaux. Le cache peut alors être renouvelé à chaque modification de la base de données.

2 - La mise en place d'un Load Balancing sur les micro services permettra de ne pas surcharger les demandes et permettra d'obtenir des réponses plus rapides.

3 - Les machines sur lesquelles seront déployé les différents micro services devront avoir des caractéristiques supérieures aux capacités actuelles en local (processeur i7 et RAM de 16 Go).