HYPOTHESE DE VALIDATION DE PRINCIPE

Imamou Chaima
Architecte logiciel MedHead+

Table des matières

I. Hypothèse n°1

- *que plus de 90 % des cas d'urgence sont acheminés vers l'hôpital compétent le plus proche du réseau :*

Le personnel médical peut saisir dans la plateforme qui leur est dédiée, le lieu de l'incident et un hôpital lui est proposé en fonction de la spécialisation et de la disponibilité de lit.

A l'exemple :

ET un patient nécessitant des soins en cardiologie.
QUAND le personnel sélectionne la spécialisation « cardiologie » ET que l'urgence est localisée à Bordeaux ALORS l'hôpital de Bordeaux va être proposé.

1. Les exigences de cette hypothèse

- Fournir une API RESTful qui tient les intervenants médicaux informés en temps réel.

- 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.

- S'assurer que le PoC est entièrement validé avec des tests d'automatisation.

- 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 la stratégie de test.

2. Méthodologie

- Mise en place du micro-service gestion des hôpitaux qui permet de rechercher et de proposer un hôpital selon la spécialisation et le nombre de lits disponibles.

- Chaque service est développé et déployé indépendamment des autres micro-services. Ainsi, chacun peut fonctionner (ou dysfonctionner) sans affecter les autres.

- Mise en place de pipelines d'intégration et de livraison continue (CI/CD) pour les micro-services.

- Mise en place de test automatisé avec l'outil JUnit.

II. Hypothèse n°2

- *que le temps moyen de traitement d'une urgence passe de 18,25 minutes (valeur actuelle) à 12,00 minutes (valeur souhaitée)*

Le partage des informations a permis de gagner en temps sur la redirection des urgences. En effet, les micro-services développés communiquent les uns avec les autres en utilisant des API REST, qui sont utilisés pour relier les micro-services en eux. Chaque service gère un seul rôle afin de gagner en efficacité donc en temps.

Les échanges hors ligne dans l'ancienne architecture, qui rallongé le temps de traitement ont été supprimé en laissant la place au partage d'information entre les différents services, qui permet de réduire considérablement le temps de traitement des demandes.

Désormais, les hôpitaux sont liés entre elles pour fournir en temps réel les informations nécessaires pour les urgences (exemple du nombre de lits disponibles).

Exemple :

Le service réservation de lit permet d'attribuer en temps réel les lits d'hôpital en fonction de la pathologie et de l'hôpital. En cas de proposition d'un hôpital, la réservation du lit va être réalisée de manière automatique. Le service réservation de lit appelle et utilise le service gestion des hôpitaux afin de pouvoir réduire le nombre de lits disponibles au moment de la réservation de lit.

1. Les exigences de cette hypothèse

- Fournir une architecture micro-service et une API RESTful qui tient les intervenants médicaux informés en temps réel.

- 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.

- S'assurer que le PoC est entièrement validé avec des tests d'automatisation .

- 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 la stratégie de test.

2. Méthodologie

- Mise en place des API REST qui permettent les échanges d'informations entre les services
- on peut adapter les micro-services en augmentant la mise à l'échelle s'il y en a besoin (scalabilité des service)

- Chaque service est développé et déployé indépendamment des autres micro-service. Ainsi, chacun peut fonctionner (ou dysfonctionner) sans affecter les autres.

- Mise en place de pipelines d'intégration et de livraison continue (CI/CD) pour les micro-services.

- Mise en place de test automatisé avec l'outil JUnit

III. Hypothèse n°3

➢ *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*

Les micro-services développés communiquent les uns avec les autres en utilisant des API REST, qui sont utilisés pour relier les micro-services en eux. Chaque service gère un seul rôle afin de gagner en efficacité donc en temps.

La mise en place d'une API Gateway est le point d'entrée unique pour les API et micro-services. En effet, elle permet d'agréger différents micro-services. L'agrégation des requêtes sera faite directement au travers de l'API Gateway, cela permet de réduire la charge réseau et la multiplication des appels.

De plus, nous mettrons en place un load balancer qui permettra d'équilibrer les charges de travail entre les serveurs en fonction de la disponibilité et des performances. Il permet de gérer la montée en charge. Le système doit être capable de s'adapter à une forte augmentation de la demande sans que son niveau de performances n'en soit impacté (Scalabilité).

Exemple :

Si le service gestion des hôpitaux est souvent sollicité, grâce à l'architecture micro-service, nous pourrons augmenter le nombre d'instances. Nous pouvons donc augmenter le nombre d'instances du service gestion des hôpitaux s'il y a trop de charges.

1. Les exigences de cette hypothèse

- Fournir une API RESTful qui sera utilisée pour relier les micro-services en eux.

- Fournir une API Gateway qui sera le point d'entrée unique pour les API et micro-services.

- 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.

- S'assurer que le PoC est entièrement validée avec des tests d'automatisation.

- 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 la stratégie de test.

2. Méthodologie

- Mise en place des API REST qui permettent les échanges d’informations entre les services
- Mise en place des API Gateway qui fournira un point d’entrée unique pour les appels http rest
- on peut adapter les micro-services en augmentant la mise à l’échelle s’il y en a besoin (scalabilité des service)
- Chaque service est développé et déployé indépendamment des autres micro-services. Ainsi, chacun peut fonctionner (ou dysfonctionner) sans affecter les autres.
- Mise en place de rapports sur les méthodes CI/CD pour le micro-service gestion des hôpitaux
- Mise en place de test automatisé avec l’outil JUnit .

IV. Hypothèse n°4

- *que la mise en œuvre explique les normes qu’elle respecte et pourquoi*

Pour le respect de la confidentialité des données, nous avons utilisé des données factices pour la mise en place du PoC et pour la réalisation de ce projet.

1. Les exigences de cette hypothèse

- Les données réelles des patients doivent à tout moment rester conformes aux réglementations européennes, notamment le RGPD.
- La confidentialité des données des patients doit être respectée et tous les prototypes ou Spike non-conformes à la production doivent anonymiser les données ou utiliser des données factices.

V. Hypothèse n°5

- *que les instructions pour mettre en production la PoC sont fournies*

Nous utiliserons la méthode agile qui prévoit une planification de mise en production par petites étapes, également appelées sprints. Pour le PoC, nous utiliserons l'outil Jira pour la gestion des mises en production.

De plus, les micro-services développés doivent être testés et déployés. Nous utiliserons des outils comme JUnit et Jenkins pour les tests d'automatisation. Des rapports seront fournis à cet effet (exemple rapport de couverture des test).

1. Les exigences de cette hypothèse

- Mise en place de la méthodologie agile.
- S'assurer que votre PoC est entièrement validée avec des tests d'automatisation.
- 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 la stratégie de test.

2. Méthodologie

- Mise en place de rapports sur les méthodes CI/CD pour le micro-service gestion des hôpitaux
- Mise en place de test automatisé avec l'outil JUnit

VI. Hypothèse n°6

- *que la mise en œuvre est terminée dans le délai imparti*

Une feuille de route a été mise en place. Nous avons utilisé la gestion de planification de la production afin de pouvoir répartir les tâches et les priorisé afin de respecter les délais de la livraison. Des points de lancement et des réunions de rétrospective ont été réalisés à chaque sprint afin de suivre l'avancement du projet.

1. Les exigences de cette hypothèse

- Respecter la feuille de route mise en place

- S’assurer que des réunions en début, en milieu et en fin de sprint sont réalisées afin de suivre l’avancement du projet.

2. Méthodologie

- Mise en place de la méthodologie agile
- Priorise les tâches mise en place
- Utilisation de l’outil Jira