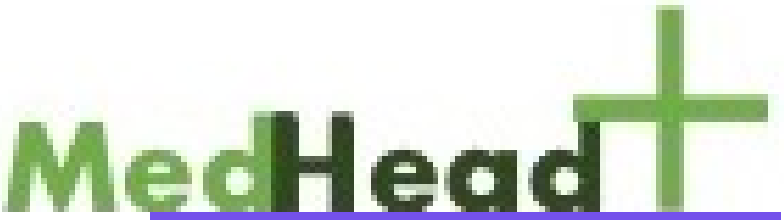
2021

Hypothèse de validation de principe



Karl Menino

Version : 1.0

12/10/2021

OPENCLASSROOMS

Table des matières

Table des matières

1. HISTORIQUE DES REVISIONS

Version	Description	Date	validateur
1.0		30/04/21	Menino karl

Ce document est l'hypothèse de validation des principes. Il énumère toutes les hypothèses de notre POC et explique comment nous avons valider ces hypothèses

Karl menino,

Architecte Logiciel,

MedHead.

2. 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
- ● 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.

3. 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.

4. 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’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.
- ● 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.

5. 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

6. Validation de l’hypothèse de la POC

Nous avons fourni cinq fonctionnalités dont l’une d’elle permet de connaitre l’hôpital le plus proche dans un rayon définit par l’utilisateur. Cette fonctionnalité fournit à plus de 90% l’hôpital le plus proche et son adresse sur l’ensemble du territoire français DOMTOM compris en cas d’urgence.

Grace à nos fonctionnalités implémentées, le processus de traitement d’une urgence devrait être raccourcie. En effet, nous donnons l’hôpital le plus proche à vol d’oiseau estimant que chaque ambulance est équipée d’un GPS. De plus, vue les règles de conduite différente des ambulance (ils ont le

gyrophare), il n'est pas pertinent d'implémenter notre propre GPS (ils peuvent prendre les sens interdit par exemple). Le temps de recherche de l'hôpital le plus proche est inferieur à 200 ms ce qui devrait raccourcir le temps de traitement en urgence d'un patient en réduisant le temps de recherche par les ambulanciers de l'hôpital le plus proche.

Le temps de réponse devrait être inferieur à 200 ms pour 800 Req/s. Cela peut être vérifier grâce à GATLING par exemple. En sachant que si nous déployons sur AWS et que nous nous assurons d'avoir un serveur suffisamment puissant, nous devrions facilement attendre ce temps de réponse. Effectivement, nous avons utiliser l'Api Géocode de google qui nous assure un temps de réponse le plus court de toutes les API disponibles pour obtenir les coordonnées GPS du patient. Ensuite nous calculons grâce à un algorithme la distance la plus courte (en tenant compte de la courbure de la terre) entre les hôpitaux proches (qui ont leur coordonnées GPS dans la BDD) et le patient. Cette méthode nous permet d'optimiser le temps de réponse de notre application plus efficacement que si nous avions utiliser l'API Distance Matrix de google et cela nous permet aussi d'optimiser le coût d'utilisation de l'API en évitant l'envoie de multiple requête à google. Le coût de notre solution est de 5$ pour 5000 requêtes.

Nous respectons les normes de sécurité OWASP. En effet, l'intégration de Springs Security nous permet d'avoir une note de A en sécurité sur la plateforme SONNARCLOUD. Bien que la sécurité CSRF est désactivé pour faciliter le test de notre application, il peut tout à fait être remis en place lors de l'application final et de la mise en développement de l'interface de notre Api Rest. En effet, par soucis de facilité lors de l'utilisation de Postman, nous l'avons désactivé car c'était contraignant de mettre le token X-XSRF-TOKEN à chacune de notre requête post. Nous respectons la RGPD en permettant aux utilisateurs de supprimer leur données grâce à une fonctionnalité DELETE. Pour terminer, nous avons mise en place une clef API et l'utilisation obligatoire du protocole HTTPS.

Les instructions de mise en production sont fournies dans les Solution building block. Ce sera un déploiement sur AWS.

Enfin, nous avons terminé dans les délais impartis.

7. Validation du comportement de la POC

A. Introduction

Nous avons 5 fonctionnalités dans notre POC :

- Recherche de l'hôpital le plus proche : Nous recherchons l'hôpital le plus proche en fonction de l'urgence en question et du rayon
- Prise de RDV : nous recherchons l'hôpital le plus proche et renvoyons ses informations en fonction de la pathologie voulue.
- Information Patient : Nous renvoyons l'historique du patient en fonction de son numéro de sécurité social.
- DELETE patient. Nous supprimons les infos du patient en fonction de son numéro de sécurité social.
- Localisation patient. Pour connaitre les coordonnées GPS du patient.

B. Recherche de l'hôpital le plus proche

Cette fonctionnalité est destinée aux urgences absolues. Nous recherchons l'hôpital le plus proche en fonction du nombre de lit disponible selon l'adresse du patient, le rayon voulu, et la pathologie demandé. La pathologie urgence regroupe tous les cas où le patient a son pronostic vital d'engager. Pour l'instant, nous n'avons pas implémenter la réservation des lits car cela demande une application déployée dans tous les hôpitaux afin de gérer la disponibilité des lits. A terme, cette fonctionnalité sera utilisée par des Thread en parallèle afin d'optimiser le temps de réponses de l'application en cas d'urgence.

C. Prise de RDV

Cette fonctionnalité est destinée à prendre des rendez-vous non-urgent. Il permet de donner l'hôpital le plus proche en fonction de la pathologie. Il devrait renvoyer les informations de l'hôpital afin que le patient ou les infirmiers prennent rendez-vous avec l'hôpital par téléphone. Nous n'avons pas intégré les téléphones des hôpitaux car nous n'avons pas cette information sur le fichier FINESS et ils devront être intégrer par les hôpitaux eux-mêmes. Les coordonnées du patient sont enregistrées dans la base de données si elle n'existe pas. Cette fonctionnalité sera utilisée par un thread en séries car son temps de réponse n'a pas d'importance et permettra de conserver des threads pour la fonctionnalité « recherche de l'hôpital le plus proche ».

D. Information Patient

Cette fonctionnalité a pour but de donner l'historique du patient en fonction de son numéro de sécurité social afin de permettre aux personnelles de santé d'aider au diagnostic du patient. Le numéro de sécurité social étant propre à chacun, nous nous assurons de n'avoir pas de doublons. L'authentification à l'appli ayant été mise en place (Spring Security) cela nous permet de respecter les normes RGPD et cela nous assure que seules les personnes autorisées peuvent avoir accès à ces informations confidentiels. De plus, le haut niveau de sécurité de l'application (Spring Security, Clef Api et l'utilisation de HTTPS) nous assure de la sécurité de ces informations.

E. DELETE patient

Cela nous permet de supprimer un patient de la base de données en fonction de son numéro de sécurité social. Cette fonctionnalité est nécessaire afin de respecter le règlement général de protections des données (RGPD).

F. Localisation Patient

Cette fonctionnalité est une fonctionnalité purement technique. Elle nous donne la localisation GPS en fonction de l'adresse. Elle est présente que dans le but de tester le temps de réponse et la disponibilité de l'API Géocode de Google.

G. Conclusion

Le comportement de notre POC couvre les hypothèses de comportement sauf pour la réservation des lits. La raison pour laquelle nous n'avons pas implémenter la réservation de lits est que cela demande une autre application qui soit déployer dans l'ensemble des hôpitaux français et qui gère les demandes de réservations et la gestion des lits. Nous avons quand même intégré les lits disponibles afin de permettre une recherche en fonction du nombre de lit par pathologie disponible. Bien que pour le peuplement de notre base de données nous ayons utilisé les données FINESS pour les hôpitaux ayant des urgences disponibles, les autres spécialités ont été simulées et ne correspondent pas à la réalité (entre autres la gestion des spécialités est non-pertinente pour les DOM-TOM qui n'ont généralement qu'un seul hôpital disponible dans un rayon de 1000 KM).

Nous pouvons quand même dire que le comportement de notre POC est largement suffisant. En effet, elle démontre que nous pouvons améliorer la qualité des traitements d'urgence donc de sauver plus de vies et, en plus, elle

est facile d'utilisation. Elle est utilisable sur l'ensemble du territoire français et, pour les urgences, donne des résultats conforme avec la réalité du terrain avec un temp de réponses inférieur à 200 ms.

8. Validation des exigences convenues de la POC

Nous avons fourni une application RestFull grâce notamment à spring-boot-starter-web et son Annotation @RestController.

Nous nous sommes assurées que toutes les données du patient soient correctement protégées grâce à Spring Security, l'utilisation d'une clef API et L'utilisation du protocole HTTPS. La note de sécurité sur SONARCLOUD est de A soit la note maximale.

L'ensemble de notre application est entièrement validé 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. Cela a été rendue possible grâce à l'utilisation de JUNIT, MOCKITO, JACOCO, GATLING, GITLAB et SONARCLOUD. Nous pouvons générer des rapports SUREFIRE et JACOCO et grâce à GITLAB nous pouvons automatiser la gestion de ces tests à chaque PUCH sur le git. La couverture de nos tests est de 55%.

La POC est facilement partageable grâce à GITLAB et GITHUB. Nous avons fourni un pipeline d'intégration et de livraison continue grâce à GITLAB. Notre stratégie de test est documentée dans le fichier « Stratégie de Test ».

Étant donné que notre POC est une API REST, il peut tout à fait s'intégrer dans une architecture micro-service et être utiliser comme un jeu de modules de construction pour d'autres modules.