MedHead

POC - Reporting

Elodie OLLIVIER, Architecte Logiciel

1. INTRODUCTION

Ce document définit l'ensemble des résultats du POC ainsi que les justifications technologiques de celui-ci.

Le but de ce document est d'assurer une couverture de test raisonnable des fonctions afin d'assurer un bon fonctionnement de la plateforme. Les choix d'implantations sont nombreux, mais le fonctionnement doit correspondre aux attentes et critères d'un produit MedHead.

2. OBJECTIFS

Le projet final sera considéré comme réussi si :

- 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 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 respecte les normes imposées
- la mise en œuvre est terminée dans le délai imparti

Les exigences suivantes ont été convenues lors de la définition de l'hypothèse de développement du POC :

- Fournir une API RESTful qui renvoie le lieu où se rendre :
  - La technologie Java est imposée par le consortium
  - L'API doit pouvoir s'inscrire à terme dans une architecture microservice
- Fournir une interface graphique qui consomme l'API :

    - Une simple page permettant de sélectionner une spécialité et de saisir la localisation est suffisante
    - Le consortium impose d'utiliser l'un des frameworks Javascript/Typescript courant du marché : Angular, React, VueJS

- S'assurer que toutes les données du patient sont correctement protégées
- S'assurer que votre PoC est entièrement validée avec des tests reflétant la pyramide de tests (tests unitaires, d'intégration et E2E) :
    - L'API doit être éprouvée 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)
- S'assurer que les équipes de développement chargées de cette PoC sont en mesure de l'utiliser comme un bloc de construction pour d'autres modules ou tout du moins comme un modèle à suivre
- La documentation technique de la PoC sera formalisée dans un fichier readme.md respectant le format markdown et contiendra au minimum :
    - les instructions pour l'exécution des tests
    - le fonctionnement du pipeline
    - le workflow Git retenu (ce dernier sera détaillé pour qu'il soit réutilisable par les équipes)
- Un document de reporting sera rédigé indiquant :
    - une justification des technologies utilisées
    - une justification du respect des normes et des principes (exemple : norme RGPD, principe d'architecture microservice, etc.)
    - les résultats et les enseignements de la PoC

3. TECHNOLOGIE

3.1 Stack

3.1.1 Front End

Comme demandé par le consortium, un framework Javascript/Typescript courant du marché a été utilisé. Nous avons choisi Angular prenant en considération la richesse et rigueur de son écosystème, d'autant plus qu'il suit le pattern MVC et possède une meilleure gestion des dépendances. De plus, Angular possède une structure à base de composants ce qui les rend hautement réutilisables et simplifie le processus de développement. L'intégration d'Angular est préconstruite avec Spring Boot.

3.1.2 Back End

L'API doit s'inscrire à terme dans une architecture microservice : les applications sont divisées en une série de services interconnectés fondés sur les capacités opérationnelles, ainsi chacun peut fonctionner (ou dysfonctionner) sans affecter les autres. Ils communiquent entre eux via, par exemple, des requêtes HTTPS à leurs APIs. Ils sont distribués et faiblement couplés, de sorte que les modifications apportées par une équipe n'affectent pas l'ensemble de l'application. L'API a donc été construite autour de deux microservices.

La technologie Java étant imposée par le consortium, nous avons préconisé l'utilisation du framework Java "Spring Boot", qui optimise :

- la gestion des dépendances
- la configuration
- la gestion des propriétés
- le monitoring
- la gestion du programme et le déploiement.

Spring Boot œuvre pour la simplification du développement des projets avec Spring Framework et se prête parfaitement au développement de microservices :

- La gestion des dépendances est simplifiée grâce aux starters qui regroupent plusieurs dépendances et homogénéisent les versions.
- L'auto configuration permet de se concentrer sur le code métier, et simplifie énormément la mise en œuvre des composants Spring qui sont utilisés.
- La gestion efficace des propriétés rend l'application configurable.
- Permet de créer rapidement des API Rest solides selon une architecture de code respectant le modèle MVC
- Intégration de Spring Security, qui permet de préconfigurer et de personnaliser des fonctions de sécurité au sein d'une application Java, notamment l'authentification et les autorisations.

3.1.3 Base de données

Pour la base de données nous avons fait le choix de PostgreSQL :

- La documentation de PostgreSQL est plus dense et complète et de façon général le support est meilleur ;
- PostgreSQL ne « dépend » pas d'un contributeur principal privé comme c'est le cas pour MySQL avec Oracle ;
- PostgreSQL propose une licence MIT qui est plus permissive que la licence GPL de MySQL.

3.2 Outils

Catégorie de test	Outils	Commentaires
Tests unitaires et d'intégrations	Junit, AssertJ, Mockito, PostgreSQL Container, Docker	JUnit est un framework de test unitaire pour le langage de programmation Java. Mockito permet de créer des objets tests. AssertJ est une bibliothèque communautaire open source utilisée pour écrire des assertions fluides et riches dans les tests Java. Le démarrage de Docker sera nécessaire pour les tests d'intégration afin de pouvoir utiliser une bdd de test avec PostgreSQL container.
Tests unitaires/d'intégration (automatisation)	Gitlab	Gitlab est un des outils permettant de créer des pipelines CI/CD (intégration et distribution continue). Un pipeline CI/CD utilise la surveillance et l'automatisation pour améliorer le processus de développement des applications, en particulier lors des phases d'intégration et de tests ainsi que pendant la distribution et le déploiement.
Tests UI/UX	Cypress	Cypress est un outil d'automatisation de test IHM (Interface graphique) open source qui dispose d'une communauté active et réactive.
Tests de montée en charge	JMeter	Apache JMeter est un projet de logiciel libre permettant d'effectuer des tests de performance d'applications et de serveurs selon différents protocoles ainsi que des tests fonctionnels. Cet outil peut être utilisé par du personnel fonctionnel.

4. NORMES ET PRINCIPES

Principe B1 : Continuité des activités des systèmes critiques pour les patients

A la portée du POC ont été effectué les tests de performance et de stress. Les résultats sont disponibles plus loin dans ce document au point 6. TESTS DE PERFORMANCE.

Principe B2 : Clarté grâce à une séparation fine des préoccupations

Comme selon les exigences du consortium, l'API s'inscrit dans une architecture microservices.

Des bases de données avec des données patients et des données hospitalières fictives ont été créées ne disposant pas des accès aux bases existantes. Il faudra prendre en compte le raccordement du service de localisation des hôpitaux au service données patients cible et au service données des hôpitaux cible.

Principe B3 : Intégration et livraison continues

A la portée du POC, un pipeline de compilation et de test a été implémentée avec Github actions.

Il sera bien évidemment possible de raccorder ces pipelines au projet cible, voire de migrer ces pipelines vers d'autres services de ci/cd le cas échéant (Jenkins, Travis, etc.).

Principe B4 : Tests automatisés précoces, complets et appropriés

Les tests unitaire, d'intégration et E2E ont été implémentés.

Les tests E2E devront se lancer depuis l'IDE, les tests unitaires et d'intégrations sont intégrés aux pipelines GitHub actions et se lancent automatiquement à chaque push vers la branche « main ».

Principe B5 : Sécurité de type « shift-left »

A la portée du POC a été implémenté Spring Security, ainsi qu'une gestion de compte avec token de connexion. Afin de protéger les données en transit et au repos hors POC, plusieurs solutions devront être mises en place : utilisation de HTTPS pour toutes les communications, pour sécuriser les données en transit, et le cryptage pour toutes les données sensibles au repos, utiliser des technologies standard pour crypter toutes les données sensibles le plus tôt possible et les décrypter le plus tard possible.

5. TESTS APPLICATIFS

Voici tous les scénarios des fonctionnalités du logiciel et les combinaisons de fonctionnalités du logiciel qui seront testés.

5.1 Liste des fonctionnalités

1. Inscription et Profil Patient :
    - Création d'un compte utilisateur et login, enregistrement des données patient.
2. Recherche d'Hôpitaux :
    - Rechercher les hôpitaux les plus proches offrant la spécialité médicale recherchée.
3. Disponibilité des Lits :
    - Vérifier la disponibilité des lits dans chaque hôpital pour la spécialité médicale sélectionnée.
4. Informations sur les Hôpitaux :
    - Fournir des informations détaillées sur chaque hôpital
5. Réservation de Lit :
    - Permettre au patient de réserver un lit dans l'hôpital choisi, en fonction de la disponibilité.
6. Sécurité et Confidentialité des Données :
    - Mettre en place des protocoles de sécurité stricts pour garantir la confidentialité des données médicales des patients.
7. Accessibilité :
    - Concevoir la plateforme pour qu'elle soit accessible aux personnes handicapées.

8. Conformité Réglementaire :

- S'assurer que la plateforme respecte toutes les réglementations et les normes médicales locales et internationales.

5.2 Tests des composants

5.2.1 Module ms-users

a. Controller

Test n°	Détails du test	Composant(s) mobilisé(s)
1	Etant donné un appel sur l'API de connexion Lorsque l'utilisateur est reconnu Alors un jeton JWT est renvoyé	Module ms-users BDD ms-users
2	Etant donné un appel sur l'API de connexion Lorsque l'utilisateur n'est pas reconnu Alors un Json d'erreur est renvoyé	Module ms-users BDD ms-users
3	Etant donné un appel sur l'API de récupération des informations d'un patient Lorsque celui-ci est reconnu Alors l'objet Patient est renvoyé	Module ms-users BDD ms-users
4	Etant donné un appel sur l'API de récupération des informations d'un utilisateur existant Lorsque celui-ci est reconnu Alors l'objet Utilisateur est renvoyé	Module ms-users BDD ms-users
5	Etant donné un appel sur l'API d'enregistrement des informations patient Lorsque le formulaire est valide Alors les informations du patient sont enregistrées	Module ms-users BDD ms-users

b. Service

Test n°	Détails du test	Composant(s) mobilisé(s)
1	Etant donné un appel sur la fonction signUpUser() Lorsque celle-ci reçoit un objet SignUpRequest Alors elle enregistre l'utilisateur en base et créée un nouveau patient avec l'email de l'utilisateur	Auth Service User Repository
2	Etant donné un appel sur la fonction getRegisteredUser() Lorsque celle-ci reçoit en objet un email Alors elle recherche l'utilisateur associé à cet email et le renvoi s'il existe	Auth Service User Repository
3	Etant donné un appel sur la fonction existsByEmail() Lorsque celle-ci reçoit en objet un email Alors elle recherche l'utilisateur associé à cet email et le renvoi un booléen selon le résultat	Auth Service User Repository
4	Etant donné un appel sur la fonction getPatient() Lorsque celle-ci reçoit en objet l'id utilisateur Alors elle recherche le patient associé à cet id_utilisateur et le renvoi s'il existe	Patient Service Patient Repository
5	Etant donné un appel sur la fonction findPatientByEmail() Lorsque celle-ci reçoit en objet un email Alors elle recherche le patient associé à cet email et le renvoi s'il existe	Patient Service Patient Repository

6	Etant donné un appel sur la fonction setUpPatient() Lorsque celle-ci reçoit un objet RegisterInfoRequest Alors elle recherche le patient associé à l'email fourni dans l'objet donné et mets à jour ses informations depuis l'objet donné	Patient Service Patient Repository
7	Etant donné un appel sur la fonction addressWithNumber() Lorsque celle-ci reçoit en objets : un objet json et l'adresse du patient Alors elle formate si besoin l'adresse du patient pour contenir le mot « street »	Patient Service
8	Etant donné un appel sur la fonction addressWithoutNumber() Lorsque celle-ci reçoit un objet json et une adresse Alors elle formate si besoin l'adresse du patient pour contenir le mot « street »	Patient Service
9	Etant donné un appel sur la fonction registerPatientInfo() Lorsque celle-ci reçoit un objet RegisterInfoRequest Alors elle met à jour les informations du patient, formate son adresse, géolocalise l'adresse et renvoie le patient	Patient Service Patient Repository

5.2.2 Module ms-localize

a. Controller

Test n°	Détails du test	Composant(s) mobilisé(s)
1	Etant donné un appel sur l'API de récupération de la liste des groupes de spécialités Lorsque l'utilisateur est authentifié Alors la liste des groupes de spécialités est renvoyée	Module ms-localize BDD ms-localize
2	Etant donné un appel sur l'API de récupération de la liste des spécialités selon un groupe Lorsque l'utilisateur est authentifié Alors la liste des spécialités du groupe est renvoyée	Module ms-localize BDD ms-localize
3	Etant donné un appel sur l'API de localisation de l'hôpital le plus proche avec un lit de disponible dans la spécialité demandée Lorsque l'utilisateur est authentifié Alors la liste des hôpitaux est renvoyée	Module ms-localize BDD ms-localize
4	Etant donné un appel sur l'API de réservation de lit Lorsque l'utilisateur est authentifié Alors un lit dans l'hôpital donné est réservé	Module ms-localize BDD ms-localize

b. Service

Test n°	Détails du test	Composant(s) mobilisé(s)
1	Etant donné un appel sur la fonction getSpecialitiesBySpecialityGroupId() Lorsque celle-ci reçoit un objet specialityGroupId Alors elle renvoie la liste des spécialités faisant parties du groupe donné	Speciality Service Speciality Repository
2	Etant donné un appel sur la fonction getSpecialitiesBySpecialityGroupName() Lorsque celle-ci reçoit un objet specialityGroupName Alors elle renvoie la liste des spécialités faisant parties du groupe donné	Speciality Service Speciality Repository
3	Etant donné un appel sur la fonction getAllSpecialityGroups() Alors elle renvoie la liste des groupes de spécialités	SpecialityGroup Service SpecialityGroup Repository
4	Etant donné un appel sur la fonction findHospitalWithinPerimeter() Lorsque celle-ci reçoit un carré de coordonnées géographiques (latitude gauche et droite, longitude gauche et droite) ainsi que la spécialité recherchée Alors elle renvoie la liste des hôpitaux ayant la spécialité demandée et au moins un lit disponible	Hospital Service Hospital Repository
5	Etant donné un appel sur la fonction bookBed() Lorsque celle-ci reçoit l'id d'un hôpital Alors elle renvoie l'hôpital avec un lit disponible en moins	Hospital Service Hospital Repository

6	Etant donné un appel sur la fonction getNearestHospital() Lorsque celle-ci reçoit un objet patientSearchRequest et un nombre locationSearchPerimeterMeters Alors elle renvoie la liste des hôpitaux les plus proches triée du plus proche au plus loin à l'intérieur du périmètre donné	Graphhopper Service Hospital Service Hospital Repository
7	Etant donné un appel sur la fonction findHospitalsWithinPerimeter() Lorsque celle-ci reçoit le périmètre de recherche en mètres, les latitude et longitude du patient, et l'id de la spécialité recherchée Alors elle crée un carré de coordonnées géographiques (latitude gauche et droite, longitude gauche et droite) ET renvoie la liste des hôpitaux ayant la spécialité demandée et au moins un lit disponible	Graphhopper Service Hospital Service Hospital Repository
8	Etant donné un appel sur la fonction routing() Lorsque celle-ci reçoit une instance Graphhopper, une latitude, une longitude, et une liste d'hôpitaux Alors elle renvoie la liste des hôpitaux donnée triée du plus proche au plus loin	Graphhopper Service Hospital Entity
9	Etant donné un appel sur la fonction run() Alors elle renvoie une instance de l'objet Graphhopper	Graphhopper Service

5.3 Tests E2E

Test n°	Détails du test	Composant(s) mobilisé(s)
1	Etant donné un patient enregistré et authentifié sur la plateforme Lorsque l'utilisateur est sur la page de recherche d'hôpitaux Alors sur le formulaire seuls sont visibles son adresse et la liste des groupes de spécialités	Module ms-localize Module ms-users Application front
2	Etant donné un patient enregistré et authentifié sur la plateforme Lorsque l'utilisateur est sur la page de recherche d'hôpitaux et a choisi un groupe de spécialité Alors sur le formulaire apparaît la liste des spécialités	Module ms-localize Module ms-users Application front
3	Etant donné un patient enregistré et authentifié sur la plateforme Lorsque l'utilisateur est sur la page de recherche d'hôpitaux et a choisi un groupe de spécialités et une spécialité et valide les critères de recherche Alors la liste des hôpitaux les plus proches avec la spécialité demandée et au moins un lit de disponible apparaît	Module ms-localize Module ms-users Application front
4	Etant donné un patient enregistré et authentifié sur la plateforme Lorsque l'utilisateur a fait une recherche et que la liste des hôpitaux est renvoyée Et que l'utilisateur clique pour réserver un lit Alors une pop-up apparaît informant l'utilisateur que la réservation est effectuée	Module ms-localize Module ms-users Application front

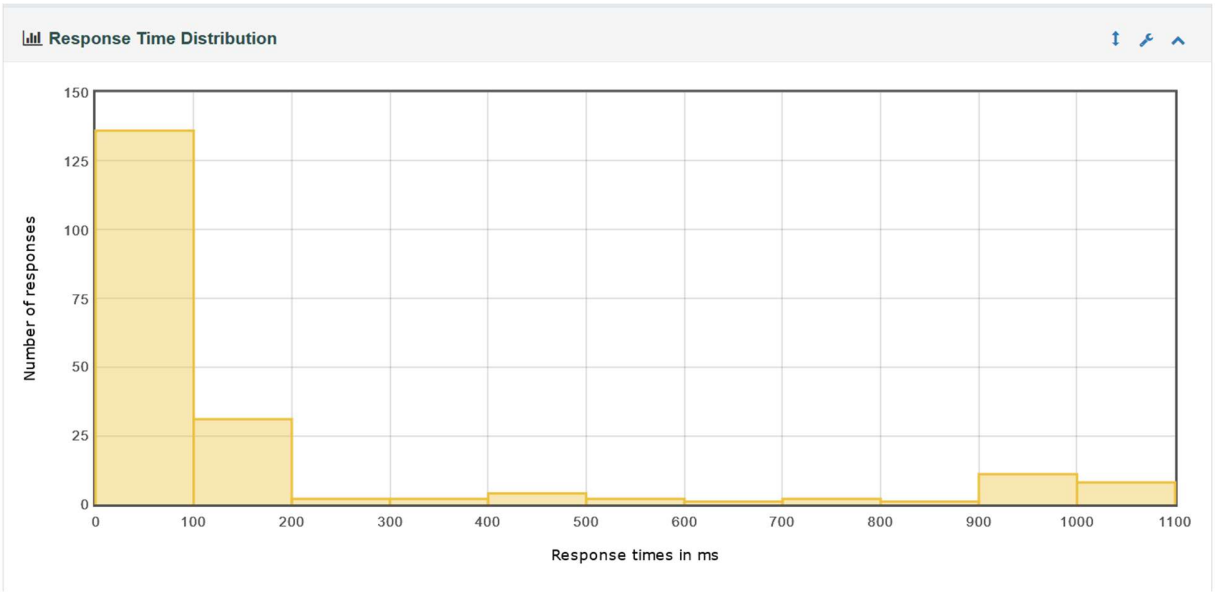
6. TESTS DE PERFORMANCE

Les tests de performance ont été réalisés avec Jmeter. Les résultats ci-dessous sont à titre purement indicatif et ne doivent pas être retenus car effectués depuis un poste local. Avec le poste local le nombre d'utilisateurs maximum pouvant effectuer les recherches d'hôpitaux en moins de 200ms est de 200. A partir de 250 le temps imparti maximum de 200ms est dépassé.

6.1 Test 200 utilisateurs

Statistics

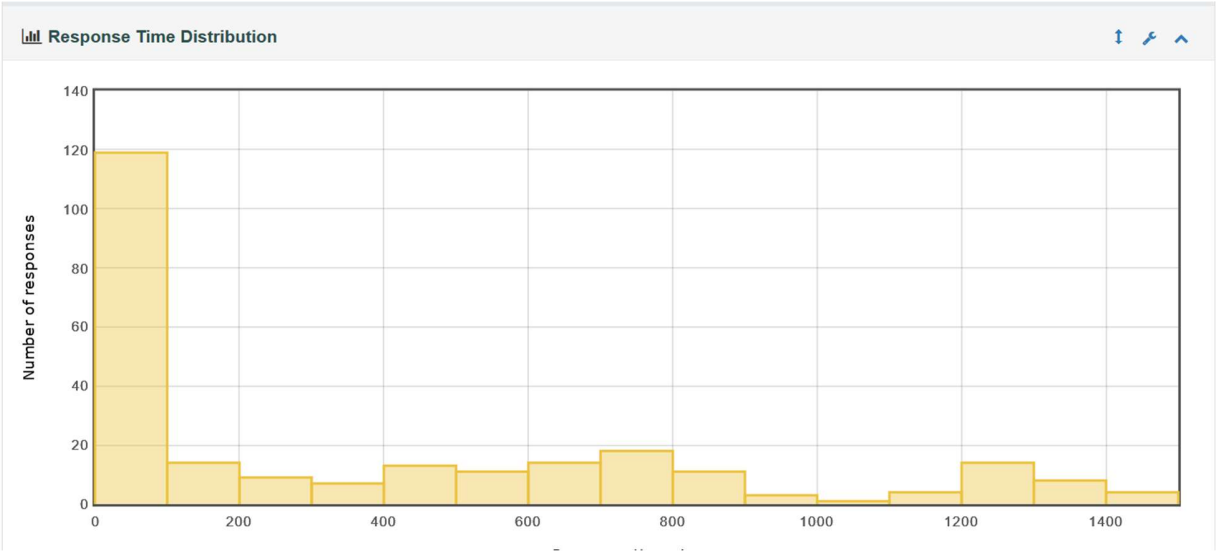
Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**200**	**0**	**0.00%**	**183.74**	**29**	**1048**	**60.50**	**869.90**	**992.95**	**1023.93**	**189.39**	**425.58**	**49.75**
search hospital	200	0	0.00%	183.74	29	1048	60.50	869.90	992.95	1023.93	189.39	425.58	49.75



6.2 Test 250 utilisateurs

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**250**	**0**	**0.00%**	**395.02**	**29**	**1455**	**115.00**	**1221.90**	**1288.25**	**1425.96**	**163.40**	**367.17**	**42.92**
search hospital	250	0	0.00%	395.02	29	1455	115.00	1221.90	1288.25	1425.96	163.40	367.17	42.92



7. RESULTATS SONAR & CYPRESS

7.1 SONAR

Toutes les notes sont à « A », toutes les pipelines sont validées par Sonar.

7.2 CYPRESS

Le parcours utilisateur a été retracé par les scénarios cypress, les tests sont validés.

8. POC ET APRES

Comme nous l'avons dans le point 6. TESTS DE PERFORMANCE on ne peut aujourd'hui démontrer le support du nombre d'utilisateurs demandé dans les circonstances présentes. Il faudra lancer les tests sur une VM avec des caractéristiques isométriques à l'environnement de production cible.

Concernant la sécurité, elle est ici minimale. Spring Security est implémenté, ainsi qu'une gestion de compte avec token de connexion. Afin de protéger les données en transit et au repos, plusieurs solutions devront être mises en place :

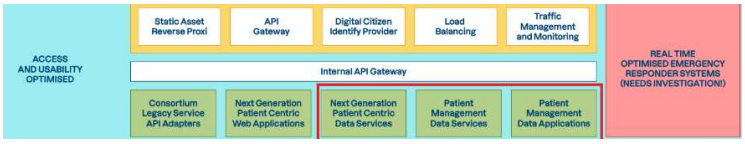
- Idéalement, imposer l'utilisation de HTTPS pour toutes les communications, pour sécuriser les données en transit, et le cryptage pour toutes les données sensibles au repos

- La meilleure stratégie consiste à utiliser des technologies standard pour crypter toutes les données sensibles le plus tôt possible et les décrypter le plus tard possible

A ce jour la clé API qui permet la connexion à Graphhopper est en dur dans le code. Il s'agit en effet d'un POC, sans VM, et cela ne doit absolument pas être le cas en production ou dans un autre environnement déployé. La clé devra figurer dans un fichier d'environnement et non stocké sur le code applicatif.

La sécurisation des conteneurs, est également à prendre en compte, et est un moyen de réduire la surface d'attaque et les risques. Une analyse régulière de la sécurité et de la vulnérabilité des conteneurs aidera à identifier les risques.

Une base de données avec des données patients fictives a été créée afin de tester la création de compte et mettre en place la sécurité d'accès avec token. Il faudra prendre en compte le raccordement du service de localisation des hôpitaux au services données patients cibles.



Enfin, le déploiement d'une surveillance continue permettra de détecter et de traiter les risques de sécurité en temps utile. Pour cela, il existe une large gamme de solutions de surveillance de services, par exemple Prométhéus.

9. APPROBATIONS

Signataire	Date	Signature
Anika Hansen		
Kara Trace		
Chris Pike		
Équipe d'intégration NHS		