

18 May 2024

CLÉMENT MARCEL

REPORTING

POC

1. Contexte	3
2. Hypothèses	3
3. Méthodologie	3
4. Objectifs du POC	3
5. Principes architecturaux	3
6. Définition des mesures clés	4
7. Collecte des données	4
8. Technologies	5
a. Back-End	5
b. Front-End	5
c. Pipeline CI/CD	5
9. Résultats	6
a. Stratégie de test	6
b. Résultats des tests	6
c. Test de stress	6
d. Test E2E	7
10. Recommandations	8
11. Conclusions	8

1. CONTEXTE

La PoC vise à développer une application web permettant aux utilisateurs de trouver des hôpitaux proches offrant des services médicaux spécifiques. L'objectif est de valider l'architecture technique et de démontrer la faisabilité et l'efficacité de la solution proposée.

2. HYPOTHÈSES

- **Performance** : La plateforme doit être suffisamment rapide et réactive pour répondre aux besoins des utilisateurs.
- **Valeur ajoutée** : La nouvelle solution doit apporter une valeur ajoutée significative par rapport aux solutions existantes.
- **Rétention** : Les utilisateurs doivent continuer à utiliser la plateforme sur une longue période.

3. MÉTHODOLOGIE

La méthodologie adoptée pour cette PoC est le "prototypage rapide", permettant de créer rapidement un modèle fonctionnel afin de recueillir des retours et de valider le projet.

4. OBJECTIFS DU POC

L'objectif principal est d'améliorer la qualité des soins aux patients. En combinant les forces de chaque entreprise du consortium, la plateforme partagée vise à soutenir l'innovation des solutions centrées sur le patient et à améliorer les activités quotidiennes des soins.

5. PRINCIPES ARCHITECTURAUX

1. **Modularité** : Utilisation d'une architecture microservices pour permettre la modularité et la scalabilité.
2. **Séparation des préoccupations** : Distinction claire entre les couches frontend, backend et données.

3. **Scalabilité** : Capacité à scaler indépendamment chaque microservice en fonction de la charge.
4. **Sécurité** : Mise en place de mesures de sécurité pour protéger les données des utilisateurs.

6. DÉFINITION DES MESURES CLÉS

Métrique	Technique de mesure	Valeur cible	Justification
Urgences acheminées	Chronométrage	12 minutes	Réduire le temps de traitement des urgences
Temps de réponse	Monitoring	< 200 ms	Assurer une haute réactivité de la plateforme
Nombre de requêtes/s	Monitoring	Jusqu'à 800	Supporter un haut volume de trafic
Taux de conversion	Google Analytics	Augmenter de 10%	Améliorer l'engagement utilisateur
Taux de rebond	Google Analytics	Réduire de 20%	Accroître l'interaction des utilisateurs avec le contenu
Temps de chargement	PageSpeed Insights	< 3 secondes	Optimiser l'expérience utilisateur

7. COLLECTE DES DONNÉES

Les données collectées incluent la position géographique, la spécialité NHS sélectionnée par le patient et les mesures de taux d'échec. En conformité avec la RGPD, aucune donnée sensible ou personnelle n'est stockée par l'application web.

De plus l'API est sécurisé par une clé nécessaire pour communiquer avec cette dernière.

Dans une future version, il est préconiser d'ajouter une double authentification afin de sécuriser davantage l'application et ses utilisateurs.

8. TECHNOLOGIES

a. Back-End

- **Langage** : Java (Imposé)
- **Framework** : Spring Boot
- **Base de données** : H2 Database

b. Front-End

- **Langage** : TypeScript
- **Librairie** : React - Choisi pour sa flexibilité, sa performance et sa large adoption dans le développement front-end moderne.

c. Pipeline CI/CD

- **Outil** : CircleCI
- **Étapes** :
 1. Backend : Exécution du build du backend, les tests se lancent automatiquement avant le build, le tout sous un environnement Linux
 2. Frontend : Execution des tests unitaires puis exécution du build du frontend sous un environnement Linux

The screenshot displays a CircleCI pipeline status. At the top, the branch is 'main' with commit 'd4fdd5a update readme'. The pipeline 'build_and_deploy' is shown as 'Success' with a green checkmark. Below, the 'Jobs' section lists two completed jobs: 'build_hospital_service' (131) and 'build_frontend' (132), both marked with green checkmarks. The repository 'OC_P11_Code' is also visible with commit '96'.

9. RÉSULTATS

a. Stratégie de test

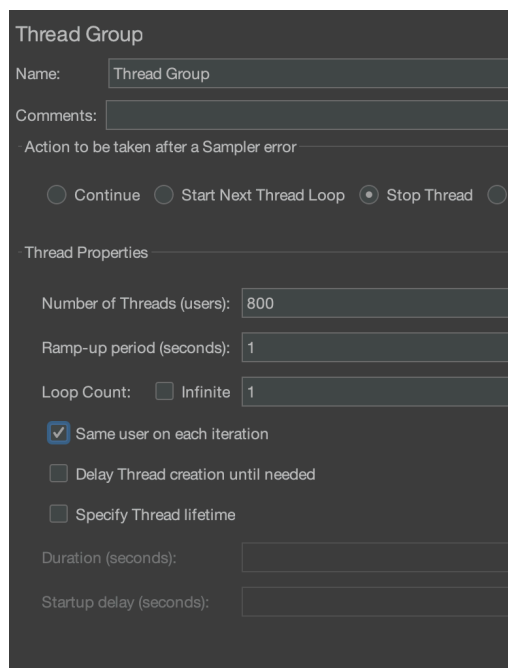
- **Tests unitaires** : Valident les fonctionnalités de base de chaque comportement.
- **Tests d'intégration** : Vérifient les interactions entre plusieurs composants.
- **Tests end-to-end** : Vérifient le fonctionnement global de l'application dans un scénario réaliste.

b. Résultats des tests

- **Unitaires** : Tous les tests unitaires passent avec succès, validant le comportement.
- **Intégration** : Les tests d'intégration montrent que les composants communiquent correctement entre eux.
- **End-to-end** : Les tests E2E valident les fonctionnalités principales du point de vue de l'utilisateur.

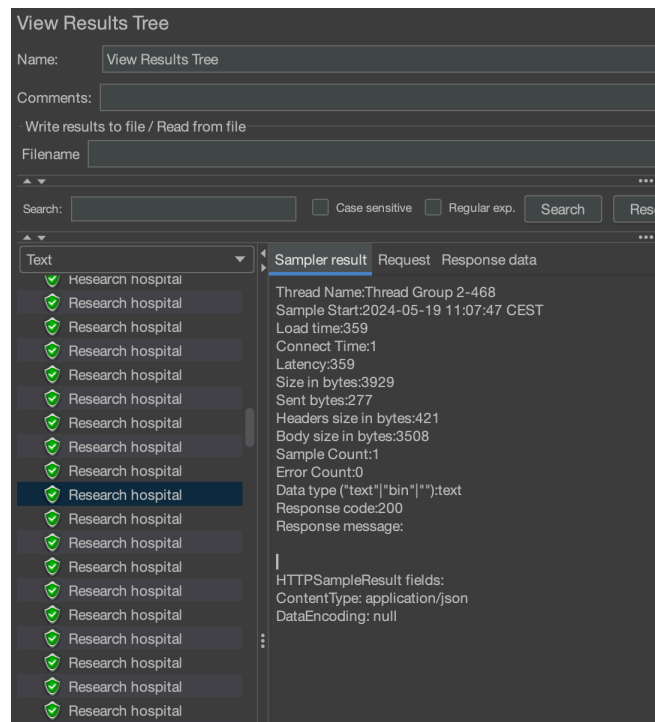
c. Test de stress

- **Objectif** : Évaluer la capacité de l'application à gérer des charges importantes.
- Configuration :



The screenshot shows the 'Thread Group' configuration window in JMeter. The 'Name' field is set to 'Thread Group'. The 'Comments' field is empty. Under 'Action to be taken after a Sampler error', the 'Stop Thread' radio button is selected. In the 'Thread Properties' section, 'Number of Threads (users)' is set to 800, 'Ramp-up period (seconds)' is 1, and 'Loop Count' is set to 1 with the 'Infinite' checkbox unchecked. The 'Same user on each iteration' checkbox is checked, while 'Delay Thread creation until needed' and 'Specify Thread lifetime' are unchecked. The 'Duration (seconds)' and 'Startup delay (seconds)' fields are empty.

- **Résultats** : L'application a montré une bonne scalabilité, avec des temps de réponse acceptables sous des charges élevées, légèrement plus lent que ce qui est attendu, du à l'environnement de développement.



d. Test E2E

Les tests End-to-End réalisés avec Cypress montrent qu'un utilisateur peut rechercher un hôpital, sélectionner une spécialité, réserver un lit et annuler la réservation. Voici les étapes principales :

Recherche d'un hôpital :

1. Saisie de la localisation et sélection de la spécialité.
2. Vérification de l'affichage du résultat de recherche.

Vérification des informations de l'hôpital :

Confirmation de la visibilité des détails de l'hôpital affiché.

Le lit est bien réservé si, le nombre de lit est mis à jour.

10. RECOMMANDATIONS

Pour améliorer les résultats de la PoC, il est recommandé de l'exécuter dans un environnement de test plus performant.

L'intégration de Docker pour les applications front-end et back-end est également suggérée.

L'intégration docker permettra de rajouter à la CI un build d'image docker et envoi sur le cloud afin de déployer l'image.

11. CONCLUSIONS

La PoC a démontré la faisabilité de l'application de recherche d'hôpitaux en utilisant une architecture microservices. Les technologies choisies se sont révélées adaptées, offrant de bonnes performances et une scalabilité suffisante. Le respect des normes RGPD a été assuré, et les tests end-to-end ont validé les fonctionnalités principales.