MedHead

Document de reporting POC

Sommaire

1. Contexte

MedHead est un regroupement de grandes institutions médicales œuvrant au sein du système de santé britannique et assujetti à la réglementation et aux directives locales (NHS). Les organisations membres du consortium utilisent actuellement une grande variété de technologies et d'appareils. Ils souhaitent une nouvelle plateforme pour unifier leurs pratiques.

Le comité d'architecture de MedHead souhaite que la plateforme pallie les risques liés au traitement des recommandations de lits d'hôpitaux dans des situations d'intervention d'urgence.

La preuve de concept (PoC) concernée par ce document, a pour but de convaincre le comité de la validation de l'architecture cible retenue.

2. Justification

2.a Technologies utilisées

Bases de données

Les données à stocker sont les suivantes :

- Hôpitaux (Nom, Position géographique, Spécialités)
- Liste des différentes spécialités médicales

Les données à stocker sont très simple et peuvent facilement être stocker dans une base de données SQL comme dans une base de données NoSQL. Pour ce POC, une base de données NoSQL a été utilisé car elle va permettre de mettre à jour les schémas de données dynamiquement, ce qui va faciliter l'évolution de l'application ou même une adaptation de ce POC pour de la production.

De plus, les bases de données NoSql permettent un scaling horizontal qui va pouvoir gérer un trafic plus important que le scaling vertical des bases de données SQL.

API Restful (Back)

Pour l'API RESTful le consortium impose l'utilisation du langage Java.

Quand il est question de développement web en Java difficile de ne pas penser au Framework Spring. Etant donné que Spring est le leader des Framework Java, il est le plus stable et le plus testé de tous. Ce qui en fait un outil très fiable pour notre projet, que ce soit au niveau de la sécurité ou de la stabilité.

De plus, Spring possède des librairies qui permettent l'ajout de fonctionnalités comme des couches de sécurité ou la gestion de base de données.

Application web (front)

Pour l'interface graphique qui consomme l'API le consortium impose l'utilisation d'un Framework Javascript/Typescript courant du marché (Angular, React, VueJS).

Etant donné que React est le Framework que le développeur du POC connaissait le mieux, ce choix a permis d'accélérer le processus de développement de ce dernier tout en respectant les conditions établies par le consortium.

Yarn a été utilisé comme gestionnaire de paquet car il est plus sécurisé, fiable et performant que NPM (le gestionnaire de paquet par défaut de NodeJS).

Workflow git

Afin de rester cohérent avec le repository existant pour ce projet (repository d'architecture) et pour rester en adéquation avec les connaissances du développeur du POC, le repository de code a également été créé sur GitHub.com.

Etant donnée que les repositories sont hébergé sur GitHub, la pipeline CI/CD utilise les GitHub Actions pour tester et build les différentes parties du POC.

Déploiement

Les GitHub Actions utilisent les fichiers de configuration docker pour créer des containers docker qui pourront facilement être déployer sur des serveurs pour une mise en production. Etant donné que nous sommes actuellement dans un POC les GitHub Actions se contentent de tester le build de ces containers.

2.b Respect des normes

2.b.1 Normes juridiques

L'application de la POC respecte la RGPD car elle ne stocke pas d'information sur ses utilisateurs.

L'application respecte la CDPA (Copyright Designs and Patents Act 1988), une loi du royaume uni qui stipule qui permet de s'assurer que les sites ne violent pas les droits d'auteur de par leurs contenus.

2.b.2 Normes techniques

L'application de la POC respecte le style d'architecture REST, les communications entre client et serveur se font en accord avec le protocole HTTP.

Le back de l'application respecte le pattern d'une application micro-services.

Le front de l'application utilise le Framework ReactJS qui suit la norme ECMAScript 6 qui est un standard pour les langages de types script, il assure l'interopérabilité des pages web à travers les différents navigateurs web.

2.c Respect des principes architecturaux

Comme on peut le remarquer dans la partie 2.b de ce document l'architecture de la POC est en adéquation avec le Principe A3 (conformité aux lois et au règlement) et le Principe C3 (Normes reconnues pour garantir les meilleurs pratiques).

La POC est en adéquation avec le principe B2 (Clarté grâce à une séparation fine des préoccupations) grâce à l'architecture micro-service pour le backend.

Les workflows git mise en place avec l'application de la POC permettent une adéquation avec les principes suivants :

- Le Principe B3 (Intégration et livraison continues) car les workflows permettent de déclencher la pipeline CI/CD lors d'évènement sur le repository GitHub, comme des push sur la branche main.
- Le Principe B4 (Tests automatisés précoces, complets et appropriés) car les workflows permettent également de déclencher automatiquement les tests et rapportent des erreurs en cas d'échec de test.

Le principe C4 (Favoriser une culture de “learning” avec des preuves de concept et des prototypes) est respecté car l’application du POC est une implémentation centrée sur l’apprentissage qui réduit les risques et valide les hypothèses.

3. Résultats

Cette section sert de validation pour la preuve de concept, tout d'abord voici un rappel des différents objectifs et facteurs de réussite que la preuve de concept doit atteindre et le résultat obtenu :

Objectifs 1 :

- **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 micro service.

Résultats :

L'application de la preuve de concept est munie d'une API RESTFUL renvoyant l'hôpital où se rendre.

L'API utilise la technologie Java qui a été imposé par le consortium et peut être intégré à une architecture micro-service à terme.

Objectifs 2 :

- **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 ;

Résultats :

L'application de la preuve de concept est munie d'une interface graphique permettant de consommer l'API, cette interface graphique suit les prérogatives du consortium en matière de technologie car elle utilise le langage Framework React et le langage Javascript. L'interface graphique possède une page permettant de sélectionner une spécialité et de saisir la location.

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

Objectifs 3 :

- **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.

Facteur de réussite :

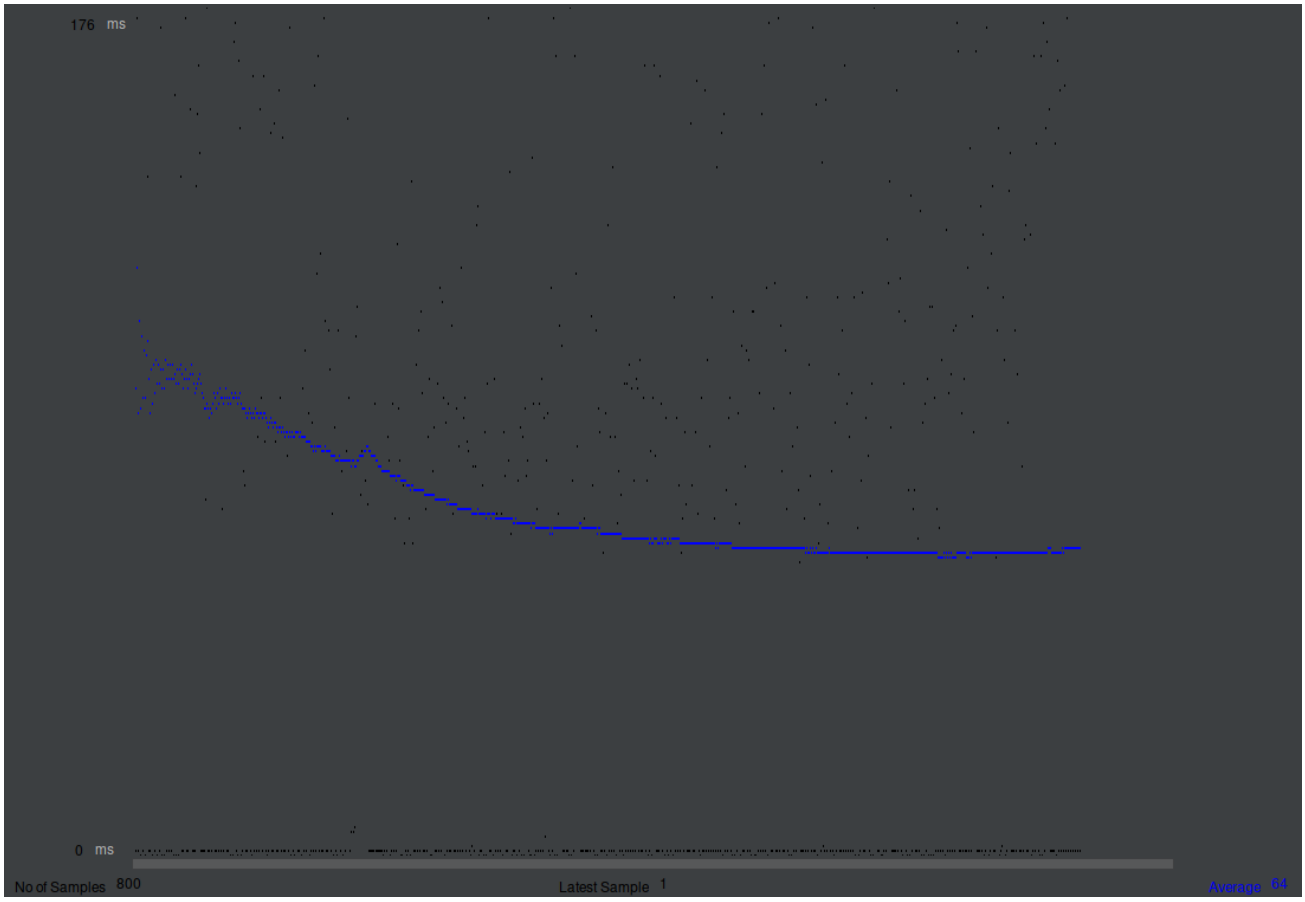
- ● nous obtenons un temps de réponse de moins de 200 millisecondes avec une charge de travail allant jusqu'à 800 requêtes par seconde, par instance de service.

Résultats :

Des tests automatisés grâce au workflow git permettent de s'assurer de la validité de la PoC à chaque fois que le code est mis à jour.

Quant au facteur de réussite, l'outil JMeter à permit d'estimer que dans plus de 90% des cas l'application répond en moins de 200 millisecondes (64 millisecondes en moyenne) avec une charge de travail de 800 requêtes par seconde.

L'image ci-dessous est un graphique représentant les temps de réponse de 800 requêtes dans une seconde. Les points noirs représentent une requête terminée et la ligne bleue représente la moyenne tout au long du test.



Les tests étant réalisés sur des machines de développement, beaucoup moins puissantes que des serveurs de production, nous pouvons facilement estimés qu'en production le temps de réponses sera en dessous des 200 millisecondes dans 95% des cas.

De plus, nous sommes dans le cadre d'une preuve de concept, le temps accordés aux détails reste minime et le code peut encore être optimisé avec un peu plus de temps.

Objectifs 4 :

- **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).**
- **Le code est versionné à l'aide d'un workflow Git adapté.**

Résultats :

Le code de la preuve de concept est facilement partageable car il est disponible sur un repository GitHub. A cette adresse : https://github.com/BrianDelalex/P11-code-repository

Le code de la preuve de concept est accompagné de workflow git (GitHub Action) qui forme des pipelines d'intégration et de livraison continue grâce à des tests automatisés, des règles pour build l'application et pour créer des containers Docker qui vont rendre le déploiement de l'application très simple.

Le repository git permet également de versionner le code.

Objectifs 5 :

- **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.**

Résultats **:**

Grâce au principe de responsabilité unique (SRP) le code se présente sous la forme de bloc facilement réutilisable pour d'autre modules.

L'architecture de la partie back (micro-service) permet une intégration facile dans un autre projet.

Objectifs 6 :

- **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).

Résultats :

Le repository GitHub contient un fichier readme.md qui permet de documenter le contenu de ce dernier. On peut y retrouver :

1. Des instructions pour build et tester l'application
2. Le fonctionnement de la pipeline
3. Le fonctionnement du workflow GitHub

Objectifs 7 :

- **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.

Résultats :

Ce document a pour rôle de synthétiser les résultats de la preuve de concept et on y retrouve, une justification des technologies utilisées, une justification du respect des normes et des principes ainsi que les résultats et enseignement de la preuve de concept.

4. Recommandations

4.a Sécurité

Accès API :

Pour accéder aux routes du micro-service utilisateur, un module de sécurité permettant de gérer le partage des ressources inter-origine (CORS) a été installé et peut donc servir d'exemple pour le futur. Cependant, dans le cadre de la preuve de concept il a été configurer de sorte à être très permissif. Dans un cadre de production, il sera très important de le configurer correctement afin qu'uniquement les adresses autorisées puisse accéder aux routes de l'API.

Le module de sécurité permet également de gérer qui a le droit d'accéder aux routes, par exemple il est possible de restreindre l'accès à certaines routes aux utilisateurs identifiés ou encore aux utilisateurs possédant certains droits. Il permet également de choisir quelles routes sont accessibles, il est donc important d'autoriser uniquement les routes utilisées par l'API.

Pour simplifier les tests lors de la preuve de concept la protection contre les attaques CSRF (Cross-Site Request Forgery) a été désactivé dans la configuration du module de sécurité, il est primordial de la réactiver avant une éventuelle mise en production car les attaques CSRF sont assez courantes.

Le système de login de la preuve de concept est simplement là pour montrer sa faisabilité, mais il n'est absolument pas à utiliser tel quel. Pour qu'il soit utilisable dans un cadre de production, il faut implémenter un système de session (JWT par exemple) qui permettra de garder un utilisateur connecté. Ce système de session pourra alors être utilisé avec le module de sécurité pour gérer l'accès aux ressources.

Les mots de passe des utilisateurs sont encryptés en utilisant la fonction BCrypt. L'avantage de cet encryptage est que l'on peut configurer le nombre d'itération d'encryptage afin de renforcer la sécurité face aux attaques brute force en fonction de l'évolution de la puissance de calcul des machines actuelles.

Accès aux bases de données :

Dans la preuve de concept l'accès aux bases de données n'est pas sécurisé. Dans un cadre de production il est primordial de mettre en place un système d'authentification très sécurisé (OpenID, x.509 Certificate par exemple) pour accéder aux données.

4.b Administration

Ajout d'hôpitaux ou de spécialités :

Pour l'ajout d'hôpitaux ou de spécialités dans les bases de données il faut un administrateur de base de données qualifiés et formés sur le projet. Uniquement une personne avec ces qualifications pourra modifier les bases de données.

Pour s’assurer que la mise en place de nouveaux sets de données n’empêche pas l’application de fonctionner correctement il est important de mettre ces nouveaux sets de données dans un environnement de préproduction et de lancer tout le cycle de test (test unitaire, test end-to-end et test d’intégration).

5. Conclusion

La preuve de concept peut être considéré comme une réussite car elle remplit les exigences demandées tout en validant les facteurs de réussite. L’application de la preuve de concept laisse des briques de code réutilisables pour les étapes à venir et des enseignements quant à la démarche à suivre pour sécuriser ces briques.