MedHead

Description éléments réutilisables

MedHead

Description éléments réutilisables

1. Introduction

Ce document permet de décrire tous les éléments de construction de l'architecture de la POC réutilisable. Il tient en compte les éléments des blocs de construction métier mais aussi de l'infrastructure du sous-système.

2. Exigences convenues de la POC

Les exigences suivantes ont été convenues lors de la définition de cette hypothèse :

a) 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.

b) S'assurer que toutes les données du patient sont correctement protégées.

c) 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.

d) 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.

e) 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.

3. Outils et technologies utilisés

Le tableau ci-dessous nous présente des outils et technologies utilisé pour la réalisation du sous-système (POC), compte tenu des besoins du projet et des avantages qu'ils présentent .

Composants	Outils & Technologies	Avantages
Microservices	Spring cloud Framework	1. Développement rapide et facile d'applications basées sur Spring ; 2. La possibilité de créer des applications autonomes ; 3. Aider à intégrer directement Tomcat, Jetty ou

		Undertow dans une application ; 4. Pas besoin de configuration XML ; 5. Quantités réduites de code source ;
Accès aux données	JPA	1. Recherche automatique des déployée métadonnées 2. Configuration standardisée -Unité Persistance 3. Code normalisé de l'accès aux données, cycle de vie, et la capacité 4. Peut remplacer les annotations avec descripteur de fichie
Base de données	MySQL	1. Totalement OpenSource 2. Très belle performance 3. Multiplate-forme 4. Multi-threadé et multi-utilisateurs
	NoSQL	1. Totalement OpenSource 2. Capable de gérer un volume important de données structurées, semi-structurées et non-structurées
Intégration continue	Jenkins	1. Facile d'installation 2. Une très grande communauté d'utilisateurs 3. Plus de 1000 plugins à notre disposition C'est écrit en Java, donc peut fonctionner dans la plus part des plate-formes
Calcul de distance entre deux endroits	GoogleDistanceMatrix	Fiable

4. Les éléments de construction (Building blocks)

L'architecture cible de la POC du sous-système du projet décrite dans le document de énoncé des travaux d'architecture présente un ensemble de composants dont :

1. **Un microservice «gateway-service :** qui sert de point d'entrée à l'application (sécurité).
2. **Gestion des données** pour stockage des données.
3. **Un Microservice «hopital-service» :** qui permet chercher l'hôpital compétent le plus proche du lieu d'incident.
4. **Un microservice «reservation-service»:** qui permet de réserver un lit d'hôpital.
5. **Un microservice «config-service»:** qui permet de centraliser tous les paramètres de configuration du système.
6. **Un microservice «registre-service»:** qui contient les information sur tous les autres microservices. Chaque mircroservice s'enregistra sur le serveur Eureka et le serveur Eureka connaît toutes les applications client exécutées sur chaque port et adresse IP.

Chacun des composants de l'architecture cité ci-desous va constituer un «Building bloc» qui est réutilisable.

4.1. Microservice gateway-service

1. **Building block name :** gateway-service.
2. **Functionality provided :** Cet élément de construction permet sert d'unique point d'entrée aux différents API des microservices et on peut y centraliser la sécurité.
3. **Link to example implementation or interfaces :** https://docs.microsoft.com/fr-fr/dotnet/architecture/microservices/multi-container-microservice-net-applications/implement-api-gateways-with-ocelot
4. **Outstanding work to complete this building block :** Customiser un outil comme Kong Gateway et le déployer sur le serveur
5. **Architectural alignment :** This building block enables or reflects the following business objectives and principles :
    - **Objective 1 :** Documenter les différents microservices et implémenter l'API Gateway.
    - **Principle 1 :** Documenter en utilisant l'outil Swagger.

- **Objective 2 :** déployer l'API Gateway
- **Principle 2 :** Conteneuriser le composant avec docker.

4.2. Gestion de données

1. **Building block name :** Solution de gestion des données.
2. **Functionality provided :** permet de stocker et réaliser des opérations sur les données de l'application.
3. **Link to example implementation or interfaces : N/A**
4. **Outstanding work to complete this building block :** déployer les deux bases de données (Mysql et NoSQL).
5. **Architectural alignment :** This building block enables or reflects the following business objectives and principles :
    - **Objective 1 :** Implémenter une base de données
    - **Principle 1 :** L'utilisation de l'UML.
    - **Objective 2 :** Sécurité des données
    - **Principle 2 :** Assurer l'intégrité des données, la confidentialité et la sécurité d'accès aux données.
    - **Objective 3 :** Migration des données
    - **Principle 3 :** Choisir une stratégie de migration des données (Big-bang ou filage).
    - **Objective 4 :** Déployer les bases de données
    - **Principle 4 :** L'utilisation de docker pour Conteneuriser la base de données.

4.3. Microservice hopital-service

1. **Building block name :** hopital-service.
2. **Functionality provided :** permet de
    - Réchercher un hôpital compétant le plus proche du lieu d'incident.
3. **Link to example implementation or interfaces :** **https://github.com/SamanoCastre/hopitalService**
4. **Outstanding work to complete this building block :** N/A
6. **Architectural alignment :** This building block enables or reflects the following business objectives and principles :
    - **Objective1:** Implémenter le microservice hopital-service en tenant compte de l'API.
    - **Principle 1 :** Utiliser la méthodologie CI/CD et valider le travail par des tests automatisés.
    - **Objective 2 :** Déployer le microservice hopital-service
    - **Principle 2 :** Conteneuriser le composant dans le docker.

4.4. Microservice reservation-service

1. **Building block name :** reservation-service.
2. **Functionality provided :** permet de
    - Réserver un lit d'hôpital suite à une recherche.
5. **Link to example implementation or interfaces :** https://github.com/SamanoCastre/reservationService
6. **Outstanding work to complete this building block :** N/A
7. **Architectural alignment :** This building block enables or reflects the following business objectives and principles :
    - **Objective 1 :** Implémenter le microservice reservation-service en tenant compte de l'API.
    - **Principle 1 :** Utiliser la méthodologie CI/CD et valider le travail par des tests automatisés.
    - **Objective 2 :** Déployer le microservice reservation-service

- **Principle 2 :** Conteneuriser le composant dans le docker.

4.5. Microservice config-service

1. **Building block name :** reservation-service.
2. **Functionality provided :** permet de
   - Centraliser les paramètres de configuration.
3. **Link to example implementation or interfaces :**
https://github.com/SamanoCastre/configService
4. **Outstanding work to complete this building block :** N/A
5. **Architectural alignment :** This building block enables or reflects the following business objectives and principles :
   - **Objective 1 :** Implémenter le microservice config-service en tenant compte de l'API.
   - **Objective 2 :** Déployer le microservice config-service
   - **Principle 2 :** Conteneuriser le composant dans le docker.

4.6. Microservice registre-service

1. **Building block name :** registre-service.
2. **Functionality provided :** permet de
   - Enregistrer les informations sur tous les autres microservices.
3. **Link to example implementation or interfaces :**
https://github.com/SamanoCastre/registreService
4. **Outstanding work to complete this building block :** N/A
5. **Architectural alignment :** This building block enables or reflects the following business objectives and principles :
   - **Objective 1 :** Implémenter le microservice registre-service en tenant compte de l'API.
   - **Objective 2 :** Déployer le microservice registre-service
   - **Principle 2 :** Conteneuriser le composant dans le docker.