MedHead

# ENONCE DES TRAVAUX D'ARCHITECTURE

---

MedHead

ENONCE DES TRAVAUX D'ARCHITECTURE

# 1. Introduction

Ce document est un énoncé des travaux d'architecture pour le projet du consortium de MedHead. Il spécifie une articulation et une direction d'architecture qui permet à l'entreprise de pallier aux risques liés au traitement des recommandations de lits d'hôpitaux dans des situations d'intervention d'urgence ; un état cible d'architecture vers lequel l'entreprise doit itérer ; il décrit un processus et une approche d'architecture sur mesure qui conviennent aux structures des équipes de l'entreprise et à la topologie de son organisation.

# 2. Énoncé des travaux d'architecture

## 2.1. Requête du projet

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 une grande variété de plateformes, de technologies et d'appareils qui souhaite utiliser Java comme langage principal.

Un consortium de quatre sociétés de premier plan s'est réuni pour consolider les efforts, les données, les applications et les feuilles de route de chacune afin de développer une plateforme de nouvelle génération centrée sur le patient et capable d'améliorer les soins de base proposés - tout en étant réactive, opérationnelle en temps réel et capable de prendre des décisions dans les situations d'urgence, en prenant en compte l'ensemble des données.

L'objectif principal du projet est de développer une plateforme de services dont les systèmes communiquent grâce à des événements. Ces services ont une responsabilité unique. Ils sont également découplés entre eux et tolérants aux pannes. En résumé, ils respectent les principes d'une architecture de microservices décomposée.

## 2.1. Description et périmètre du projet

Dans le cadre du développement de cette plate-forme de nouvelle génération centrée sur le patient et capable d'améliorer les soins de base proposés, les domaines d'architecture suivants sont considérés :

- ✓ Architecture métier du système
- ✓ Architecture des données et de l'information

- ✓ Architecture de la sécurité
- ✓ Architecture technologique

Architecture s'intégrant facilement avec de nouveaux systèmes et avec les partenaires. Autrement dit, il faut gérer le changement et l'infrastructure d'exécution.

# 3. Objectifs et périmètre

## 3.1. Objectifs

Les principaux objectifs de l'entreprise sont les suivants.

1. Tirer parti de la géolocalisation pour permettre aux intervenants de réserver un lit dans l'hôpital le plus proche du lieu d'incident.
2. L'architecture devra être évolutive pour permettre à nos services de se déployer sur diverses régions à travers des villes et des pays donnés.
3. Améliorer les soins dispensés aux patients.
4. Eliminer les risques liés au système d'intervention d'urgence en temps réel.

## 3.2. Périmètre

Le projet consiste à une plateforme de nouvelle génération centrée sur le patient et capable d'améliorer les soins de base proposés - tout en étant réactive, opérationnelle en temps réel et capable de prendre des décisions dans les situations d'urgence, en prenant en compte l'ensemble des données. Une POC Proof Of Concept devant être réalisé pour prouver la faisabilité du projet. Ce POC se limite à la recherche de disponibilité dans l'hopital le plus proche du lieu d'incident en fonction de la spécialité.

## 4. Parties prenantes, rôles et résultats

Le tableau suivant montre les parties prenantes du projet, leurs rôle, et la façon dont le travail d'architecture répondra à leurs préoccupations par l'expression de plusieurs visions.

| Parties prenantes | Rôle et organisation | Résultats |
|---|---|---|
| Kara Trace | CIO Ursa Major Health | Intégration des systèmes de médecine générale en temps quasi réel avec des prestataires de soins de santé. |
| Anika Hansen | PDG, Jupiter Schelduling Inc | Architecture de domaine consolidée avec des événements sécurisés circulant en temps réel entre les systèmes |
| Équipe d'intégration des systèmes de santé du Royaume-Uni | Emergency Expert System | Enrichissement des données et accès aux données anonymisées des hôpitaux et des patients |
| Ashley Ketchum | Architecte métier exécutif, Emergency Expert Systems | |
| Chris Pike | Architecte métier principal, Schedule Shed | Réductions de pannes. Flux d'événements normalisés et infrastructure de sécurité pour réduire la responsabilité de l'organisation causée par des erreurs de planification. |
| Samano CASTRE | Architecte Logiciel | Prouver la faisabilité du projet par la réalisation d'un POC (Proof of concept) |

## 5. Responsabilités (RACI)

Un ensemble de responsabilités ont été définies. Et celles-ci seront confiées aux différents acteurs du projet. Pour la présentation des responsabilités/Rôle, on s'appuie sur la matrice RACI représentée par le tableau ci-dessous :

R.A.C.I. est un acronyme qui signifie :

1. R- Responsible : les personnes chargées de réaliser la tâche
2. A - Accountable : les autorités chargées de valider le travail
3. C - Consulted : les personnes à consulter
4. I - Informed : les personnes à informer

| R Responsible<br>A Accountable<br>C Consulted<br>I Informed | Directeur Général | Architectes métier | Architecte Logiciel | Chief Information Officer | Resp. Validation Qualification | Dév. |
|---|---|---|---|---|---|---|
| Enoncé sommaire des traveaux | A | R | | I | | |
| Document de définition d'architecture | A | R | | I | | |
| Les solutions building blocks | A | R | | I | | |
| Hypothèse | A | R | | I | | |
| Principes d'architecture | A | R | | | | |
| Réaliser un POC | | A | R | | | C |
| Hypothèse de validation | A | R | C | | | C |
| Stratégie de tests | A | R | | | I | |
| Organisation des tests | | A | | | R | C |
| Validation des tests | | A | | | R | C |
| Livraison POC | | R | | | | C |

# 6. Processus projet

## 6.1. Réunions régulières

Dans le cadre de ce projet de développement d'une nouvelle plate-forme de services un ensemble de réunions professionnelles est nécessaires, telles que : La réunion du comité de pilotage, la réunion de lancement du projet, la revue de projet, point d'avancement, et des réunions de clôtures.

## 6.2. Comité de pilotage

Pour assurer le bon déroulement des opérations en fonctions des objectifs de ce projet, nous allons créer un comité de pilotage.

Ce comité se constituera de : du CIO (directeur principal de l'information), du CPO (Chef de produit) et du responsable ingénierie.

## 6.3. Intégration continue

Intégration continue ou (CI continious Integration) regroupe un ensemble de pratiques visant à accélérer le développement d'une application tout en garantissant la qualité du code. Ces consistent à tester de manière automatisée chaque révision de code avant de le déployer en production.

De nombreux outils permettent d'assurer l'intégration continue tels que : Jenkins, Travis CI, gitlab CI,Bamboo, TeamCity, etc…

Dans le cadre de ce projet, nous allons utiliser Jenkins.

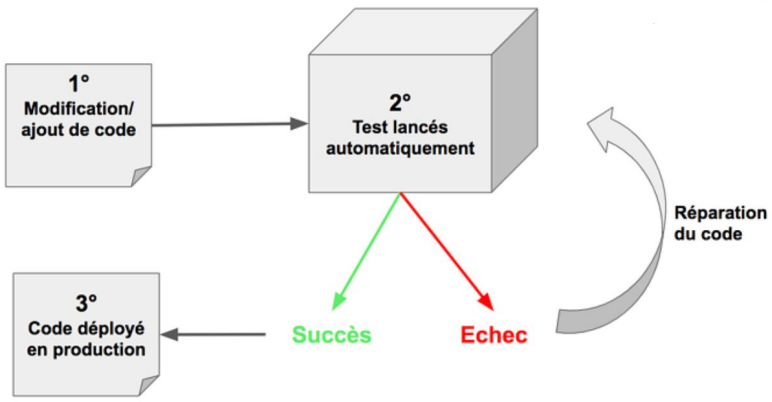


Figure 1 - illustration intégration continue

## 6.4. Procédure en cas de changement

Un processus de gestion du changement est une série d'étapes pour introduire et livrer une transformation**.**

Il existe alors 3 types de changements :

1. **Changement standard :** Changement qui représente peu de risques. Ce type de changement est pré-autorisé, voire automatisé.

2. **Changement urgent :** il s'agit ici de résoudre des erreurs, des incidents, mais aussi de s'adapter à des circonstances particulières. Il doit être mis en œuvre assez rapidement.

3. **Changement normal :** C'est le type de changement qui suit la procédure classique. Contrairement aux changements standards, il suit une chaîne d'approbation et de validation.

Pour tout changement dans le cadre de ce projet, nous allons procéder ainsi :

1. **Objectifs, Périmètre et Risques :** La règle des 7R nous permettra de définir les objectifs du changement et établir son périmètre. Cette règle consiste à se poser les 7 questions suivantes :
   a) **Requis** : qui a requis ce changement ?
   b) **Raison** : quelle est la raison de ce changement ?
   c) **Retour** : quel est le retour attendu de ce changement ? Quels sont les bénéfices attendus ?
   d) **Risque** : quels sont les risques encourus à la réalisation ou la non-réalisation de ce changement ?
   e) **Ressources** : quelles sont les ressources nécessaires pour réaliser ce changement ? De quoi aurons-nous besoin ?
   f) **Responsable** : qui est responsable de la réalisation de ce changement ?
   g) **Relation** : quelle relation avec d'autres changements ? Existe-t-il des dépendances ?

2. **Catégorisation :** Catégoriser un changement consiste à lui affecter une priorité. La méthode consiste à déterminer la priorité d'un changement grâce à une matrice prenant en compte l'impact et l'urgence de ce changement. Voir figure 2 ci-dessous.

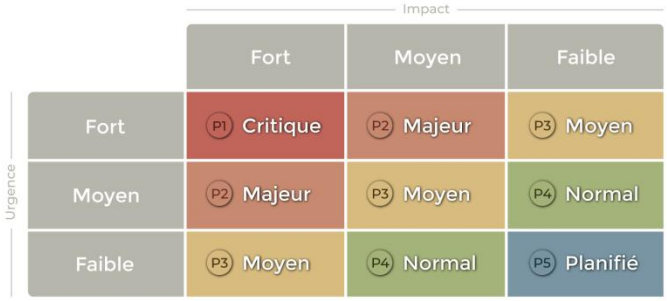


Figure 2 - Matrice Impact/Urgence

3. **Planification :** Il s'agit d'établir un calendrier de changements autorisés. Ce plan de changement doit être en fonction des besoins du business en évitant que les changements s'interfacent avec d'autres processus de changement.

4. **Plan de remédiation :** On parle aussi de Backout Plan, Il s'agit de prévoir un plan de retour dans le cas où le déploiement se passe mal. Dans le cadre de ce projet, nous allons toujours réaliser un sauvegarde du système avant d'y apporter tout changement, afin de pouvoir revenir à l'état initial.

# 7. Approche architecturale

## 7.1. Process d'architecture

*Voir document de définition de l'architecture*

## 7.1. Contenu de l'architecture

### 7.1.1 Phase A : Vision de l'architecture

a. Problèmes du système existant

Le principal problème du système actuel est le risque d'acheminement d'un patient vers un hôpital qui ne serait pas le plus proche dans la région.

### 7.1.2. Contraintes de l'architecture cible

Les contraintes de l'architecture cible est d'ordre budgétaire, temporel et techniques :

a) Les systèmes et processus existants ne doivent pas être significativement entravés pendant les phases du projet

b) La feuille de route architecturale et les orientations alimentant les exigences fonctionnelles doivent respecter le Cadre des normes numériques et technologiques du NHS.

c) Les données réelles des patients doivent à tout moment rester conformes aux réglementations européennes, notamment le RGPD.

d) Les phases initiales du projet devraient viser la création de modules de construction réutilisables ou de modèles pour des modules de construction futurs qui respectent les meilleures pratiques convenues.

### 7.1.3. Objectifs et exigences

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.

### 7.1.2. La Phase B : Architecture métier

**Phase B – Architecture Métier :**

Le métier avec ses exigences, ses processus et ses entités, gouverne l'architecture d'entreprise. Elle permet de fixer l'architecture cible et de mesurer les impacts.

Ce projet vise à fournir un état des métiers cibles. Ce projet possède une gouvernance architecturale commune au sein du Consortium, par le biais d'un groupe de gestion de la plateforme et des systèmes détenus conjointement dont le but est de réaliser des économies d'échelle et de respecter une feuille de route stratégique pour la livraison d'une plateforme qui offre aux membres du Consortium à la fois une agilité et une mise sur le

marché plus rapide, avec des niveaux de sécurité des données et de tolérance aux pannes exigés de la part des institutions du secteur de la santé.

Le diagramme ci-dessous nous présente l'architecture cible du projet.

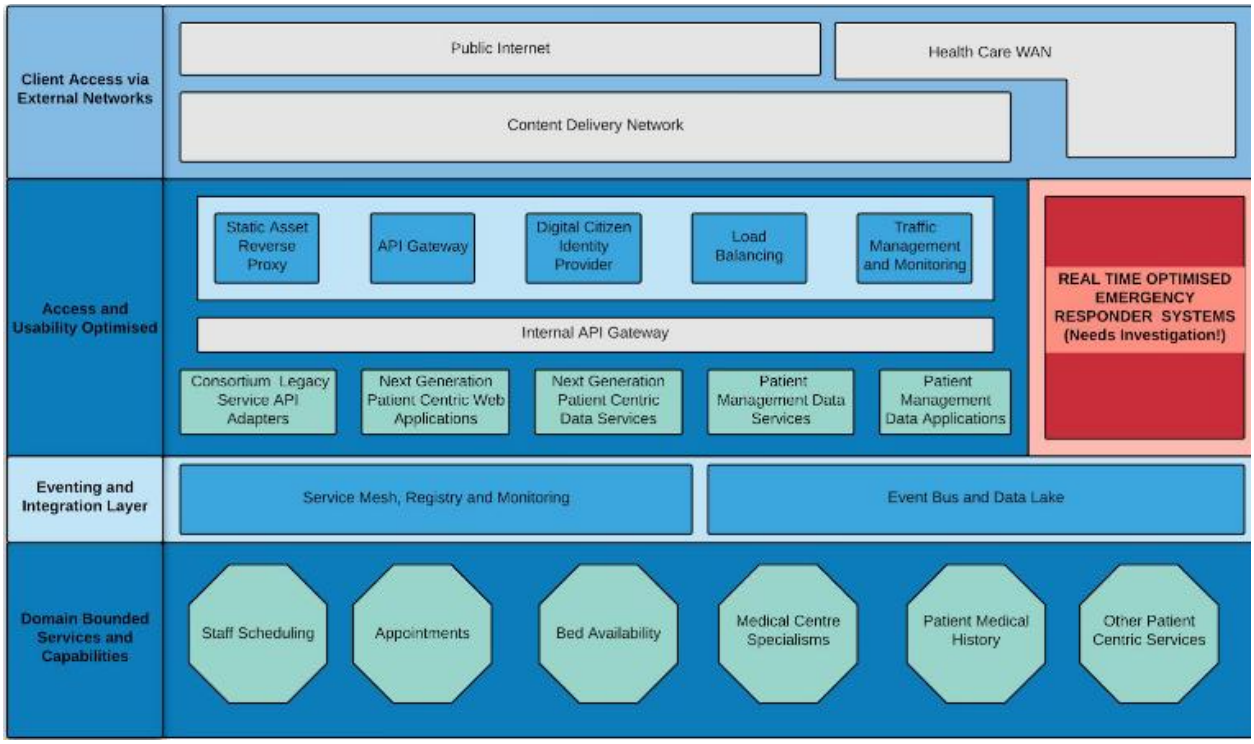


Figure 4 - Architecture cible

## Description de composants

*Voir le document de définition d'architecture*

**Analyse des exigences**

Dans cette section nous décrivons les différentes exigences ou conditions du projet. Elles sont regroupées en 2 groupes distincts : Exigences techniques et exigences opérationnelles. Le tableau ci-dessous nous présente les exigences et les solutions.

| Exigences | Solutions |
|---|---|
| **Exigences techniques** | |
| que plus de 90 % des cas d'urgence sont acheminés vers l'hôpital compétent le plus proche du réseau. | - Utilisation d'un API externe permettant de calculer la distance entre le lieu d'incident et les hôpitaux. |
| Les phases initiales du projet devraient viser la création de modules de construction réutilisables ou de modèles pour des modules de construction futurs qui respectent les meilleures pratiques convenues. | - Utilisation d'une architecture avec un couplage faible<br>- L'indépendance des modules<br>- Maintenance facile<br>- La réutilisation des composants développés |
| 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 | - Gestion des charges avec l'utilisation de load balancer. |
| Les systèmes et processus existants ne doivent pas être significativement entravés pendant les phases du projet. | - Choix d'une architecture permettant l'évolutivité du système, donc de la maintenance facile. |
| Doit permettre l'intégration des travaux des différentes équipes de la plate-forme sans créer de pannes du système | - Pratique de l'intégration continue. |
| **Exigences opérationnelles** | |
| Les données réelles des patients doivent à tout moment rester conformes aux réglementations européennes, notamment le RGPD. | - Sécurité par HTTPS :<br>- Sécurité par TLS :<br>- Mise en place d'un firewall<br>- Mise en place d'un système d'authentification sécurisé par login/mot de passe. |
| Optimisation de la fonctionnalité «recherche hopital» | - Utilisation d'une base de données NoSQL |

Au vu des objectifs et des différentes contraintes techniques et fonctionnelles du projet, telles que :

1. Gestion des charges et performance (jusqu'à 800 requête en 200 milisecondes)
2. Sécurité des données
3. Amélioration et modification future du système (évolutivité).
4. Forte croissance de l'entreprise (scalabilité).
5. Stabilité et fiabilité de la plate-forme.

L'architecture recommandée est une architecture en micro-services. En effet, l'architecture Microservices propose une solution en principe simple : découper une application en petits services, appelés Microservices, parfaitement autonomes qui exposent une API *REST* que les autres Microservices pourront consommer. Le POC sera réalisée selon ce style d'architecture.

## Architecture Systeme d'urgence en temps réel (POC)

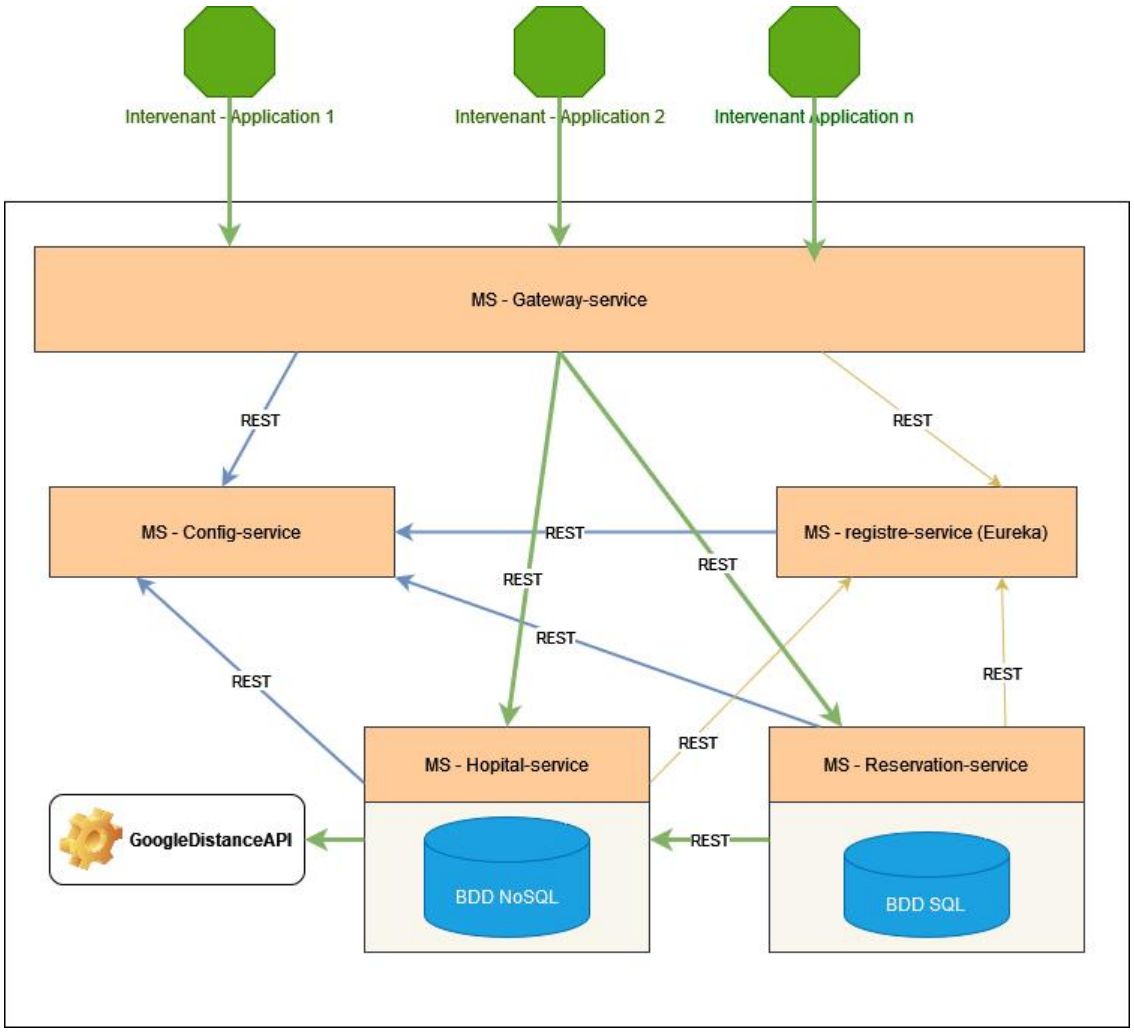


Figure 6 - Architecture système urgence (réservation hopital)

### Description des composants

Le tableau ci-dessous nous présente la liste des composants de l'architecture du sous-système de gestion d'urgence

| Composants | Description |
|---|---|
| MS Gateway-seervice | (Également appelée passerelle API) est le point d'entrée unique pour les API et microservices backend définis (qui peuvent être à la fois internes et externes). Assise devant les API, l'API Gateway agit en tant que protecteur, renforçant la sécurité et assurant l'évolutivité et la haute disponibilité. |
| MS Registre-service | Répartition de charge, est une technologie conçue pour distribuer la charge de travail entre différents serveurs ou applications. Le but : optimiser la performance globale de l'infrastructure, son rendement et sa capacité. |
| MS Config-service | Ce composant centralise la configuration de tous nos microservices |
| MS Hopital-service | Ce composant permet de rechercher l'hopital le plus proche du lieu d'incident en fonction de la spécialité demandée. |
| MS Reservation-service | Ce composant permet aux intervenants de réserver un hôpital |
| Google Distance API | Ce composant permet de calculer la distance entre le lieu d'incident et les adresses des hôpitaux présent dans la base. |
| API REST | Une API REST (également appelée API RESTful) est une interface de programmation d'application (API ou API web) qui respecte les contraintes du style d'architecture REST et permet d'interagir avec les services web RESTful. |

### Interactions entre les micro-services métiers

Le diagramme ci-dessous présente les liens de communication existant entre les différents micro-services métiers de l'architecture :

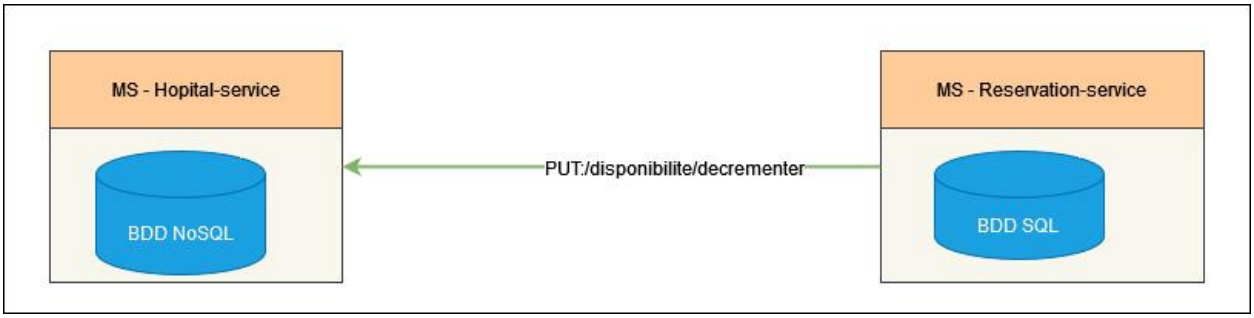


Figure 7 - Diagramme interactions entre les microservices

### Description des dépendances entre les micro-services

1. **Le micro-service « reservation-service» :** a besoin de décrémenter le nombre de lits disponible après chaque réservation (cette disponibilité se trouve dans le ms hopital-service, d'où l'interaction).

### Les endpoints des micro-services

Le tableau ci-dessous présente quelques endpoints des micro-services du projet :

| Micro-services | Endpoints | Description action |
|---|---|---|
| Hopital-service | GET:/hopital | Permet de rechercher l'hopital le plus proche du lieu d'incident |
| | GET:/specialites | Permet de retourner la liste complète des spécialités |
| | GET:/disponibilite/decrementer | Permet de décrémenter la disponibilité d'un hopital |
| Reservation-service | POST:/reservation | Permet d'enregistrer une réservation |

### Les fonctionnalités

Le tableau ci-dessous nous présente les différentes fonctionnalités du projet avec les micro-services responsables de ces fonctionnalités ainsi que les ressources exposées et nécessaires.

| Micro-services | Fonctionnalités | Ressources exposées | Ressources nécessaires |
|---|---|---|---|
| MS hopital-service | - Rechercher un hôpital disponible | - Hopital | - Lieu incident<br>- specialité |
| MS reservation-service | - Faire une réservation dans un hopital | - Réservation | - Specialite<br>- Hopital<br>- Intervenant |

### Enjeux de sécurité : protection des données

La sécurité des micro-services consiste à protéger l'intégrité des API :

1. **API GATEWAY :** On fait une authentification centralisée grâce à la passerelle d'API pour limiter l'accès aux ressources et aux API comme le montre la figure représentant l'architecture modifiée.

2. **CHIFFREMENT DES DONNES QUI TRANSITENT**
   Pour un accès plus sécurisé, on va utiliser un certificat SSL (Secure Socket Layer) ou plutôt TLS (Transport Layer Security): il s'agit d'une technologie standard destinée à la sécurité de la connexion Internet et à la protection des données sensibles qui sont transmises entre deux systèmes, empêchant les criminels de lire et de modifier les informations transférées, y compris d'éventuelles informations personnelles.

   TLS est avant tout un protocole de cryptage des données circulant sur le web. Il s'agit d'un système de sécurisation qui agit entre les serveurs et le réseau. TLS est en quelque sorte la version améliorée de SSL. C'est un protocole en deux couches, qui reprend les caractéristiques principales du SSL tout en améliorant certaines fonctions, de manière à mieux sécuriser les échanges de données. Les deux couches désignent deux clés qui agissent l'une en parallèle de l'autre lorsqu'un utilisateur souhaite par exemple rejoindre une navigation sécurisée.

   Lorsqu'un visiteur s'authentifie, il fait appel à ce certificat SSL mis en place par le serveur. Ce certificat est en quelque sorte la carte d'identité électronique du site sécurisé. Il a également pour rôle de chiffrer les données entrées par l'utilisateur afin de leur garantir un premier niveau de protection.

Ce certificat contient deux clés distinctes : d'un côté, une clé publique qui valide le certificat racine et de l'autre, une clé privée, plus adaptée au chiffrement des échanges de données afin de leur assurer un niveau de protection maximal.

## Technologies / Outils

Le tableau ci-dessous présente les outils retenus pour chaque microservice, compte tenu des avantages, inconvénients et prix.

| Composants | Technologies/outils | Avantages |
|---|---|---|
| Base de données relationnelles | ✓ Mysql | ✓ Excellente performance<br>✓ Open-source<br>✓ Léger, portable et gratuit<br>✓ Rapide sur les requêtes simples |
| Base de données non relationnelles | ✓ NoSQL (Mongodb) | ✓ Capable de gérer un volume important de données structurées, semi-structurées et non structurées.<br>✓ Offre une performance supérieure par rapport à MySQL<br>✓ Gestion des données volumineuses qui gèrent la vitesse, la variété, le volume et la complexité des données. |
| Géolocalisa-tion | ✓ GoogleMatrixAPI | ✓ Facile d'utilisation<br>✓ Payant mais une version d'essai existe<br>✓ Rapidité du chargement de la carte |
| Intégration continue (CI) | ✓ Jenkins | ✓ Open-source et gratuit<br>✓ Facile d'installation<br>✓ Une très grande communauté d'utilisateurs<br>✓ Plus de 1000 plugins à notre disposition<br>✓ C'est écrit en Java, donc peut fonctionner dans la plus part des plate-formes |
| Microservices (back-end) | ✓ Spring boot Cloud | ✓ Configuration facile<br>✓ Absolument pas de génération de code et pas |

| | | |
|---|---|---|
| | | de configuration XML requise<br>✓ Gestion des dépendances simplifiée grâce à platform-bom |

# 8. PLAN DE TRAVAIL

## 8.1. Initialisation du projet

L'initialisation de projet est la première phase du cycle de vie d'un projet.

Lors de cette phase il est essentiel de :

1. Préciser l'objectif du projet ;
2. Identifier les acteurs et les parties prenantes ;
3. Identifier les risques ;
4. Définir les délais, les ressources, la démarche, les outils de pilotage et le plan de communication ;
5. Définir les ressources humaines et financière ;
6. Définir la démarche, les outils de pilotage et le plan de communication.

**Les livrables :**

1. **Énoncé des travaux d'architecture sous format PDF**
2. **Document de stratégie de test**
3. **Document complet sur les principes d'architecture**
4. **Hypothèse de validation de principes complète**

## 8.2. Les différentes phases

Compte tenu de possibles évolutions du projet, nous avons prévu d'avoir une approche agile avec la méthode Scrum et d'utiliser la technique d'intégration continue (CI) pour éviter des régressions lors des mises à jour.

Puis, réaliser **chacune des fonctionnalités** de chaque micro-service en suivant le principe du Schema ci-dessous :

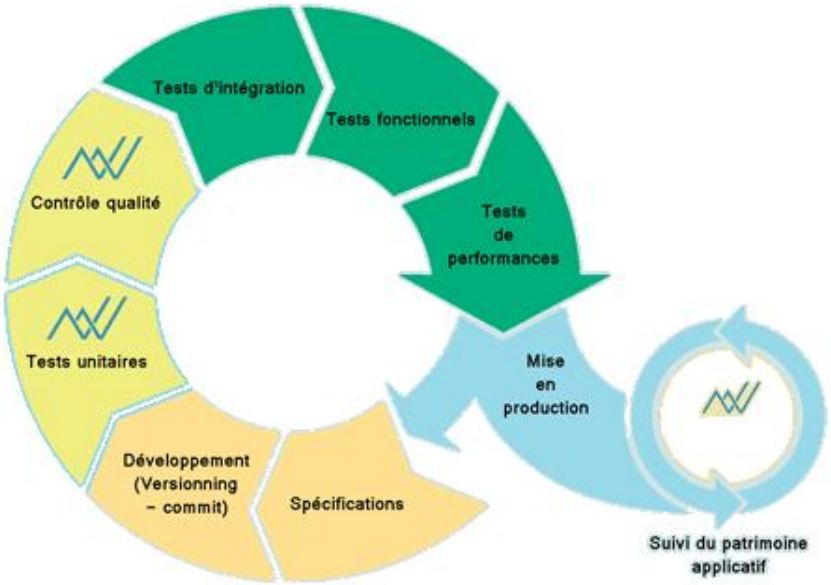


Table 1- intégration continue

Ainsi chacune des fonctionnalités de chacun de nos microservices auront à faire ce parcours :

### 8.2.1. Analyse

Cette phase consiste à définir l'objectif de cette fonctionnalité, la faisabilité et le degré de souplesse qui pourra être accordée.

### 8.2.2. Conception

Cette phase consiste à établir une liste de tâches associées à la fonctionnalité, à les ordonner tenant compte et à et à indiquer leur enchaînement logique en tenant compte des ressources disponibles et de leur charge de travail maximale

### 8.2.3. Spécifications

Cette phase consiste à définir l'ensemble d'exigences à satisfaire par la fonctionnalité.

### 8.2.4. Développement (code)

C'est la phase où les développeurs codent la fonctionnalité (Front-End et Back-End) et connectent les interfaces pour atteindre les objectifs définis.

#### 8.2.5. Tests Unitaires

C'est la phase qui consiste à s'assurer du fonctionnement correct d'une partie précise de la fonctionnalité. Elle a pour objectif d'isoler le comportement de la partie de code à tester de tout facteur extérieur et de vérifier qu'il est conforme à ce qui est attendu.

#### 8.2.6. Contrôle de qualité

C'est la phase où la conformité de la fonctionnalité développée est vérifiée. Le résultat peut être conforme, non conforme mais avec possibilité de retouches ou non conforme et tout recommencer.

#### 8.2.7. Test d'intégration

C'est la phase où on s'assure que toutes les parties développées indécemment fonctionnent bien ensemble.

#### 8.2.8. Test fonctionnel

C'est la phase où on on va tester la fonctionnalité développée via des parcours en simulant les actions de l'utilisateur (clics, saisies claviers, mouvement de souris, …).

#### 8.2.9. Test de performances (Benchmark)

C'est la phase où on mesure la performance de la fonctionnalité développé. Le test de performances  a pour objectif de mesurer le temps de réponse.

#### 8.2.10. Mise en production

Cette phase consiste à déployer la fonctionnalité entièrement testée et totalement fonctionnelle.

## 8.3. Livrables

Un ensemble de livrables seront livré au client lors du déploiement de l'application :

1. **Documentation technique** : Document qui décrit l'utilisation, la fonctionnalité ou l'architecture d'un produit, d'un système ou d'un service.
2. **Cahier de test /Recette :** Document qui regroupe de nombreux points permettant de mener à bien votre mise en production.
3. **Document de déploiement :** Document qui réunit toutes les bonnes pratiques pour la mise en service de l'application.
4. **Les bases de données** : Les différentes bases qui servent à stocker les données utilisateurs.

## 8.4. Plan de communication

Un large processus de consultation nécessite l'utilisation et la combinaison de différentes méthodes en tenant compte des caractéristiques du public cible. La stratégie de consultation est présentée dans le tableau ci-dessous pour ce Projet.

| Étape projet | Type de message | Support | Fréquence | Cibles | Responsables |
|---|---|---|---|---|---|
| -Préparation et définition du projet | - Demande de compléments d'informations.<br>- Risques Éventuels du projet | - Réunions<br>- Entrevues<br>- Courriels<br>-Téléphones | - A chaque mise à jour du projet | - Chris PIKE<br>- Daniel Anthony | - Ashley Ketchum<br>- Samano CASTRE |
| - Définition du plan d'exécution du projet | - Réunion du comité de pilotage, | - Réunions | Au début et une fois par mois selon les besoins | MOA :<br>Chris PIKE, | - Ashley Ketchum<br>-Samano CASTRE |
| - Exécution et le pilotae du projet | - Compte rendu réunion<br>- Justification de l'avancement<br>- Utilisation des ressources<br>- Résultats intermédiaires<br>- Demandes d'informations d'ordre techniques | - Courriels | Toutes les deux semaines | - MOA:<br>Chris PIKE,<br>Comité de direction | Chris PIKE |
| | - Point d'avancement<br>- Réunion de travail<br>- Revu du projet<br>- Tâches à réaliser<br>- Résultats obtenus<br>- Réunion d'équipe | - Réunions,<br>- Courriels, | Quotidien (Au fur et à mesure des mises à jour) | Équipe technique | MOE :<br>- Ashley Ketchum<br>- Samano CASTRE |
| | - Rapport de validation de toute ou partie de | - Courriels | Récurrente | Ashley Ketchum | - Ashley Ketchum |

| | | | | | |
|---|---|---|---|---|---|
| | fonctionnalités, | | | | - Testeurs |
| | - Nouvelles demandes<br>- Demandes d'évolution | - Réunions,<br>- Courriels, | Récurrente | Ashley Ketchum ,<br>Samano CASTRE | - Chris PIKE<br>- comité de direction |
| - Mise en exploitation et la clôture du projet | - Clôture COPIL<br>- Clôture projet | - Réunions | 1 Fois en fin de projet | Ashley Ketchum Comité de Direction<br>Chris PIKE | - Ashley Ketchum |

## 8.5. Analyse des risques

L'objectif d'une analyse de risques consiste à éliminer ou réduire le niveau de risques en mettant en place des mesures de prévention adéquates. Elle permet d'assurer un lieu de travail sain et sécuritaire pour tous. Dans le cadre de ce projet, nous avons identifié certains risques qui méritent d'être pris en compte.

On distingue différents types de risques :

1. **Financiers** : coût supérieur à l'estimation, manque de budget, etc.
2. **Humains** : manque de compétences, absentéisme, démission au cours du projet, conflits au sein de l'équipe, etc.
3. **Temporels** : retards ou mauvaise estimation des délais, etc.
4. **Techniques** : logiciel inadapté, pannes, matériel obsolète, etc.

Pour évaluer et gérer les risques nous avons élaboré la matrice de criticité suivante :

| Probabilité (P) | | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Certains** | **5** | 5 | 10 | **15** | **20** | **25** |
| | **Probable** | **4** | **4** | 8 | 12 | **16** | **20** |
| | **Peu probable** | **3** | **3** | 6 | 9 | 12 | **15** |
| | **Rare** | **2** | **2** | **4** | 6 | 8 | 10 |
| | **Extrêmement Rare** | **1** | **1** | **2** | **3** | **4** | 5 |
| | | | **1** | **2** | **3** | **4** | **5** |
| | | | **Mineur** | **Significatif** | **Sévère** | **Critique** | **Catastrophique** |
| | | | **Gravité (G)** | | | | |

| | **Criticité (C Min)** | **Criticité (C Max)** | **Description** |
|---|---|---|---|
| | 13 | 25 | Risque inacceptable, mesures indispensables de réduction du risque |
| | 5 | 12 | Risque à surveiller, mesures adaptées de réduction du risque |
| | 1 | 4 | Risque acceptable |

Le tableau ci-dessous nous présente la gestion des risques du projet

| Risque identifié | Causes | Conséquences /Impact | Risque Potentiel | | | Solution/Plan d'action (Barrières) | Risque Résiduel | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | P | G | C | | P | G | I |
| Le système d'intervention d'urgence en temps réel n'est pas adapté aux incidents | Non prise en compte des spécialités | Erreur lors de l'Acheminement du patient vers un hôpital | | 4 | >13 | Test de performance précoce d'une preuve de concept représentative | 1 | 4 | 4 |
| Le système d'intervention d'urgence en temps réel gère la latence concernant la disponibilité des lits des hôpitaux du réseau | Problème lié à la gestion des charges | Temps d'attente trop long, l'application peut être figée | | 4 | >13 | Test de performance précoce d'une preuve de concept représentative | 3 | 3 | 9 |
| Le système d'intervention d'urgence en temps réel n'offre pas de solution lorsqu'il n'y a pas de lits d'hôpital disponibles pour la spécialisation requise | Problème lié à la gestion de la disponibilité | Incohérence dans la proposition de l'hôpital compétent | | 4 | >13 | Attribution à l'hôpital le plus proche disposant de lits | 2 | 2 | 4 |
| Le système d'intervention d'urgence en temps réel ne répond pas dans les 200 millisecondes à la demande de lits | Problème de latence | Temps d'attente d'une réponse trop longue | | 2 | >5 | Validation de principe pour vérifier qu'un paramètre d'« urgence » est en mesure de fournir au système de réponse le nom d'un hôpital disposant d'un lit en moins de 200 nanosecondes, pendant un pic d'activité | 2 | 2 | 4 |
| Le système d'intervention | Style d'architecture | Produit inutilisable | 3 | 5 | 15 | Utiliser OpenAPI pour | 1 | 1 | 1 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| d'urgence ne peut pas être interfacé par d'autres systèmes | inadapté | | | | | définir les contrats de service. Inclure cela dans la première preuve de concept -personnaliser par la suite en tant que solution building blocks | | | |

# 9. CRITERES D'ACCEPTATION ET PROCEDURES

## 9.1. Métriques et KPIs

Le KPI c'est le suivi d'indicateurs de type ratios comparant le "prévisionnel" et le "réalisé" en termes de temps, de consommation de budget et de ressources.

Les métriques suivantes seront utilisées pour déterminer le succès de ce travail d'architecture :

| Métrique | Technique de mesure | Valeur cible |
|---|---|---|
| Acheminement patients | Pourcentage redirigé vers l'hôpital compétent le plus proche | 90% des cas d'urgence doivent être acheminé vers l'hôpital compétent le plus proche du réseau. |
| Temps moyen de traitement d'une urgence | Comparaison avec valeur actuelle | Passer de 18.25 minutes à 12 minutes |
| Performance | Temps de réponse | Moins de 200 miliecondes avec une charges de 800 requêtes par secondes, par instance de service |

## 10. APPROBATIONS SIGNEES

| Partie prenante | Fonction | Signature |
|---|---|---|
| Anika Hansen | PDG, Jupiter Scheduling Inc | |

## 11. CONCLUSION

Nous avons établi cette déclaration de travail d'architecture conformément à la méthode ADM de TOGAF pour répondre aux besoins du consortium de MedHead en matière d'architecture.