MedHead

Solution Building Blocks

Table des matières

Historique des versions

Version	Description	Date	Responsable
1.0	Création	20/01/2022	Magalie Morteau

Introduction

Ce document présente ***Solution Building Blocks*** en se basant sur la Documentation de l'Architecture de Définition.

Il fournit une description de tous les éléments de base réutilisables obtenus lors de la conception et de la mise en œuvre du POC pour la gestion du système de demande d'intervention médicales urgentes, ERS (Emergency Responder Subsystem).

Tous les modules de construction doivent être succincts et précis comme dans l'exemple ci-dessous. Vous pouvez créer des renvois vers des informations supplémentaires stockées ailleurs dans le répertoire d'architecture.

Modèle de module de construction :

1. Nom du module de construction :
2. Fonctionnalité fournie :
3. Lien vers des exemples d'implémentation ou d'interfaces :
4. Travail supplémentaire pour terminer ce module de construction :
5. Alignement architectural:
    - Objectif 1 :
    - Principe 1 :

Référence sur la documentation officielle TOGAF :

The TOGAF Standard, Version 9.2 - Building Blocks (opengroup.org)

Solution Building Blocks

Framework Java

Nom du module de construction : Solution pour créer un nouveau projet Java

Fonctionnalité fournie : Le module de construction permet de normaliser la création de projets dans le cadre de solutions.

Lien vers des exemples d'implémentation ou d'interfaces :

https://spring.io/tools

https://spring.io/projects/spring-boot

https://spring.io/projects/spring-framework

https://spring.io/projects/spring-data-jpa

Travail supplémentaire pour terminer ce module de construction : Implémenter des conteneurs Docker pour normaliser l'exécution Spring-CLI.

Alignement architectural : Ce module de construction permet ou reflète les objectifs suivants :

- S’assure que la PoC peut être facilement intégrée dans le développement futur : rendre le code facilement partageable.
- 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.

Architecture Microservices

Nom du module de construction : Solution pour l'architecture technique

Fonctionnalité fournie : Ce module de construction permet de standardiser l'architecture technique de notre logiciel. Soit une architecture en couches, avec un rôle pour chaque couche :

- La couche Controller : gestion des interactions entre l'utilisateur de l'application et l'application.
- La couche Service : implémentation des traitements métiers spécifiques à l'application ;
- La couche Repository : interaction avec les sources de données externes ;
- La couche Model : implémentation des objets métiers qui seront manipulés par les autres couches.

Lien vers des exemples d'implémentation ou d'interfaces :

https://spring.io/microservices

https://openclassrooms.com/fr/courses/6900101-creez-une-application-java-avec-spring-boot/7077993-structurez-et-configurez-votre-projet

Travail supplémentaire pour terminer ce module de construction :

- Implémentation des DTO
- Découper les services avec une couche interface et une pour son implémentation

Alignement architectural : Ce module de construction permet ou reflète les objectifs et les principes métiers suivants :

- Principe B2 : Clarté grâce à une séparation fine des préoccupations.

API Restful

Nom du module de construction : Solution pour une API Restfull

Fonctionnalité fournie : Ce module de construction permet de rendre l'API conforme à une architecture Rest.

Lien vers des exemples d'implémentation ou d'interfaces :

https://spring.io/guides/tutorials/rest/

https://spec.openapis.org/oas/v3.1.0

https://www.gekko.fr/blog/les-bonnes-pratiques-a-suivre-pour-developper-des-apis-rest

Travail supplémentaire pour terminer ce module de construction :

- Spécification OpenAPI des contrats de service
- Déploiement via une infrastructure conteneurisée, immuable et reproductible

Alignement architectural : Ce module de construction permet ou reflète les objectifs et les principes métiers suivants :

- Principe B2 : Clarté grâce à une séparation fine des préoccupations.
- Principe C3 : Normes ouvertes convenues pour garantir des normes élevées

Gestion du stockage des données

Nom du module de construction : Solution pour stocker les données

Fonctionnalité fournie : Ce module de construction permet de gérer les données persistantes sur lesquelles des opérations CRUD seront effectuées.

Les données pour la gestion des hôpitaux sont stockées dans la table : hospital

Les données pour la gestion des des pathologies sont stockées dans la table : pathology

Les données pour la gestion des lits disponibles en fonction de la pathologie sont stockées dans la table : hospital_pathology

Les données pour la gestion des demandes d'interventions urgentes sont stockées dans la table : emergency_log

Les données pour la gestion des patients sont stockées dans la table : patient

Lien vers des exemples d'implémentation ou d'interfaces :

PostgreSQL https://www.postgresql.org/

Travail supplémentaire pour terminer ce module de construction :

- Modéliser les données.
- Une base de données par service.
- Persistance des données.
- Indexation et requêtes.
- Compléter les données initialisées sur l'ensemble des territoires
- Déployer le data repository.

Alignement architectural : Ce module de construction permet ou reflète les objectifs et les principes métiers suivants :

- Principe B2 : Clarté grâce à une séparation fine des préoccupations.

Gestion des interventions urgentes

Nom du module de construction : Solution pour gérer les demandes d'intervention urgentes

Fonctionnalité fournie : Ce module de construction permet de gérer les demandes d'intervention urgentes.

Les opérations CR sur la table emergency_log : Create, Read

Lien vers des exemples d'implémentation ou d'interfaces :

https://spring.io/guides/tutorials/rest/
https://spec.openapis.org/oas/v3.1.0
https://spring.io/microservices
https://spring.io/guides/gs/spring-boot-docker/

Travail supplémentaire pour terminer ce module de construction :

- Spécification OpenAPI des contrats de service
- Déploiement via une infrastructure conteneurisée, immuable et reproductible

Alignement architectural : Ce module de construction permet ou reflète les objectifs et les principes métiers suivants :

- Fonctionnalité F1 : Retourne en temps réel les coordonnées de géo localisation et les consignes de l'hôpital le plus proche
- Fonctionnalité F2 : Retourne l'historique du journal d'intervention des urgences médicales.
- Fonctionnalité F3 : Retourne le détail d'une intervention en urgence médicale.
- 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

Gestion des lits disponibles

Nom du module de construction : Solution pour gérer les lits disponibles

Fonctionnalité fournie : Ce module de construction permet de gérer les lits disponibles en fonction d'une pathologie et d'un hôpital.

Les opérations RU sur la table hospital_pathology : Read, Update

Lien vers des exemples d'implémentation ou d'interfaces :

https://spring.io/guides/tutorials/rest/
https://spec.openapis.org/oas/v3.1.0
https://spring.io/microservices
https://spring.io/guides/gs/spring-boot-docker/

Travail supplémentaire pour terminer ce module de construction :

- Spécification OpenAPI des contrats de service
- API Rest
- Déploiement via une infrastructure conteneurisée, immuable et reproductible

Alignement architectural : Ce module de construction permet ou reflète les objectifs et les principes métiers suivants :

- Fonctionnalité F4 : Retourne la liste des hôpitaux de la zone d'intervention ayant des lits disponibles dans un service demandé.
- Fonctionnalité F5 : Réserve un lit dans un hôpital et un service.
- Fonctionnalité F6 : Retourne la liste des lits disponibles par spécialisation et hôpital.
- Fonctionnalité F7 : Retourne la liste des lits disponibles par hôpital pour une spécialisation donnée.

Gestion des distances

Nom du module de construction : Solution pour gérer les distances entre 2 points

Fonctionnalité fournie : Ce module de construction permet de calculer les distances en kms entre deux points.

Lien vers des exemples d'implémentation ou d'interfaces :

https://spring.io/guides/tutorials/rest/
https://spec.openapis.org/oas/v3.1.0
https://spring.io/microservices
https://spring.io/guides/gs/spring-boot-docker/

Travail supplémentaire pour terminer ce module de construction :

- Spécification OpenAPI des contrats de service
- Déploiement via une infrastructure conteneurisée, immuable et reproductible

Alignement architectural : Ce module de construction permet ou reflète les objectifs et les principes métiers suivants :

- Fonctionnalité F8 : Retourne la distance en kms entre 2 points géo localisés avec leurs latitudes et longitudes

Gestion des hôpitaux

Nom du module de construction : Solution pour gérer les hôpitaux

Fonctionnalité fournie : Ce module de construction permet de gérer les hôpitaux

Les opérations R sur la table hospital : Read

Lien vers des exemples d'implémentation ou d'interfaces :

https://spring.io/guides/tutorials/rest/
https://spec.openapis.org/oas/v3.1.0
https://spring.io/microservices
https://spring.io/guides/gs/spring-boot-docker/

Travail supplémentaire pour terminer ce module de construction :

- Spécification OpenAPI des contrats de service
- API Rest
- Déploiement via une infrastructure conteneurisée, immuable et reproductible

Alignement architectural : Ce module de construction permet ou reflète les objectifs et les principes métiers suivants :

- Fonctionnalité F9 : Retourne la liste des hôpitaux
- Fonctionnalité F10 : Réserve le détail d'un hôpital

CI/CD

Nom du module de construction : Solution pour gérer l'intégration et le déploiement continu

Fonctionnalité fournie : Ce module de construction permet de gérer l'intégration et le déploiement continu

Lien vers des exemples d'implémentation ou d'interfaces :
https://github.blog/2022-02-02-build-ci-cd-pipeline-github-actions-four-steps/
https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-java-with-maven
https://docs.github.com/en/actions/deployment/deploying-to-your-cloud-provider/deploying-to-amazon-elastic-container-service
https://docs.github.com/en/actions/deployment/deploying-to-your-cloud-provider/deploying-to-azure

Travail supplémentaire pour terminer ce module de construction :
- Déploiement via une infrastructure conteneurisée, immuable et reproductible

Alignement architectural : Ce module de construction permet ou reflète les objectifs et les principes métiers suivants :
- Principe B3 : Intégration et livraison continues
- Principe C2 : Référentiel d'architecture centralisé et organisé comme source de référence

Tests

Nom du module de construction : Solution pour gérer les tests

Fonctionnalité fournie : Ce module de construction permet de gérer les tests automatiques

Lien vers des exemples d'implémentation ou d'interfaces :
Junit : https://junit.org/junit5/
Postman : https://www.postman.com/
Surefire: https://maven.apache.org/surefire/maven-surefire-plugin/
Jacoco : https://www.jacoco.org/
SonarCloud: https://sonarcloud.io/
Cucumber: https://cucumber.io/
Selenium: https://www.selenium.dev/
Gatling: https://gatling.io/

Travail supplémentaire pour terminer ce module de construction :
- Environnement de Test et Recette
- Tests intégrations, et E2E
- Tests de performance, charge et stress

Alignement architectural : Ce module de construction permet ou reflète les objectifs et les principes métiers suivants :
- Principe B4 : Tests automatisés précoces, complets et appropriés
- Les applications doivent être construites à l'aide de tests automatisés qui garantissent la fiabilité à la fois fonctionnelle et non fonctionnelle de la mise en œuvre.
- TDD et Pyramide de test
- S’assurer que le 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.