

Hypothèse de développement d'une preuve de concept pour le sous-système d'intervention d'urgence en temps réel

Historique des versions

Version	Description	Date	Responsable
1.0	Création	06/12/2021	OpenClassRooms
1.1	Modification paragraphes « Méthodologie ». Ajout du paragraphe « Validation du POC ». Ajout du paragraphe « Feedback et Recommandations ». Ajout du paragraphe « Annexe » avec les copiés d'écran des résultats de test.	20/01/2022	Magalie Morteau

Déclaration d'hypothèse

Nous pensons que la mise en œuvre d'une preuve de concept pour le sous-système d'intervention d'urgence en temps réel par l'équipe d'architecture métier du Consortium MedHead permettra :

- d'améliorer la qualité des traitements d'urgence et de sauver plus de vies ;
- de gagner la confiance des utilisateurs quant à la simplicité d'un tel système.

Nous saurons que nous avons réussi quand nous verrons :

- que plus de 90 % des cas d'urgence sont acheminés vers l'hôpital compétent le plus proche du réseau ;
- que le temps moyen de traitement d'une urgence passe de 18,25 minutes (valeur actuelle) à 12,00 minutes (valeur souhaitée) ;

- 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 ;
- que la mise en œuvre explique les normes qu'elle respecte et pourquoi ;
- que les instructions pour mettre en production la PoC sont fournies ;
- que la mise en œuvre est terminée dans le délai imparti.

Exemple de comportement et description de la capacité

Le sous-système d'intervention d'urgence en temps réel est destiné à recevoir une ou plusieurs spécialités médicales (voir les [Données de référence sur les spécialités](#)) et une banque de données d'informations récentes sur les hôpitaux afin de suggérer l'hôpital le plus proche offrant un lit disponible, associé à une ou plusieurs spécialisations correspondantes. Le lieu de l'incident d'urgence doit également être fourni.

Par exemple, SUPPOSONS trois hôpitaux, comme suit :

Hôpital	Lits disponibles	Spécialisations
Hôpital Fred Brooks	2	Cardiologie, Immunologie
Hôpital Julia Crusher	0	Cardiologie
Hôpital Beverly Bashir	5	Immunologie, neuropathologie diagnostique

ET un patient nécessitant des soins en cardiologie.

QUAND vous demandez des soins en cardiologie ET que l'urgence est localisée près de l'hôpital Fred Brooks

ALORS l'hôpital Fred Brooks devrait être proposé

ET un événement devrait être publié pour réserver un lit.

Exigences convenues de la PoC

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

- 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.
- S'assurer que toutes les données du patient sont correctement protégées.
- 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.
- 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) et documenter votre stratégie de test.
- 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.

Méthodologie

La documentation et la PoC qui en résulteront seront présentées aux membres du Conseil d'administration pour décrire les enseignements tirés de la PoC. Des rapports sur les méthodes CI/ CD seront présentés au personnel technique afin d'expliquer comment mettre à jour le système.

Voir les documents suivants :

- La stratégie de test
- Le plan de test
- Le pipeline CI/CD
- Statement Of Architecture Work

Validation de l'hypothèse / Résultats

Le tableau suivant reprend les KPI de la POC qui ont été documentés dans « Statement architecture of work » avec les résultats obtenus lorsque l'indicateur de performance a été évalué. Les valeurs obtenues en vert pour les KPI performants, en orange pour ceux qui ne sont pas performants mais non bloquants pour notre POC, la solution est connue et sera implémentée. Et en rouge pour les KPI non performants et bloquants.

KPI	Justification /Notes complémentaires	Valeur cible	Valeur obtenue
Taux de couverture exigences fonctionnelles	Le taux de couverture des exigences fonctionnelles . Toutes les fonctionnalités exigées ont été implémentées avec succès.	100%	100%
Taux de réussite aux tests	Le taux de réussite de tous les tests manuels et automatiques	100%	100%
Temps de réponse	Le temps de réponse correspond au délai entre le moment où la demande de service a été effectuée et celui auquel la réponse est obtenue.	<= 200 ms	<= 25 ms
Bogues	Résultat de l'analyse qualité de Sonar sur le niveau de fiabilité du code en fonction du nombre de bogue	A	A
Vulnérabilité	Résultat de l'analyse qualité de Sonar sur le niveau de sécurité du code en fonction du nombre de vulnérabilité. 1 vulnérabilité facilement corrigeable avec l'implémentation d'un DTO.	A	D
Taux de couverture test (code ERS)	Résultat de l'analyse qualité de Sonar sur le % de lignes de code couvertes par les tests (sur les principaux composants de l'API)	>= 80%	100%
Code Smells	Résultat de l'analyse qualité de Sonar sur le niveau de maintenabilité du code en fonction du nombre de Code Smells	A	A
Durée de l'effort	La mise en œuvre est terminée dans le délai imparti. Soit 4 Sprints d'une semaine.	<= 1 mois	1 mois
Taux de réussite	Plus de 90 % des cas d'urgence sont acheminés vers l'hôpital compétent le plus proche du réseau.	>= 90%	100%

Le tableau suivant reprend la check liste des critères d'acceptation pour la POC qui ont été documentés dans « Statement architecture of work » avec les résultats obtenus lorsque le critère a été évalué.

Critère	Justification /Notes complémentaires	Valeur obtenue
Réutilisabilité	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) et documenter votre stratégie de test.	OK
Evolutivité	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 (SBB).	OK
Micro-service	Découpage en micro service permettant de tester individuellement chaque service de l'architecture. Clarté grâce à une séparation fine des préoccupations (cf. Principe B2)	OK
API RESTful	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.	OK
Pipeline CI/CD	Exécutions CI/CD génèrent des journaux ou des sorties clairs qui peuvent être analysés pour isoler les builds en échec ou les erreurs dans les étapes de build, de test et de livraison. Intégration continue OK. Livraison continue KO. Car absence d'environnement de Test et Recette pour déployer l'API et la base de données PostgreSQL. Par défaut on a automatisé la génération du JAR et sa mise à disposition pour téléchargement.	CI OK
Stratégie de Test	Tests automatisés précoces, complets et appropriés (cf. Principe B4)	OK

Liste des fonctionnalités implémentées dans le POC et testées.

ID	Fonctionnalités	OK KO
F1	Retourne en temps réel les coordonnées de géo localisation et les consignes de l'hôpital le plus proche.	OK
F2	Retourne l'historique du journal d'intervention des urgences médicales.	OK
F3	Retourne le détail d'une intervention en urgence médicale.	OK
F4	Retourne la liste des hôpitaux de la zone d'intervention ayant des lits disponibles dans un service demandé.	OK
F5	Réserve un lit dans un hôpital et un service.	OK
F6	Retourne la liste des lits disponibles par spécialisation et hôpital.	OK
F7	Retourne la liste des lits disponibles par hôpital pour une spécialisation donnée.	OK
F8	Retourne la distance en kms entre 2 points géo localisés avec leurs latitudes et longitudes.	OK
F9	Retourne la liste des hôpitaux	OK
F10	Retourne le détail d'un hôpital	OK

Liste des contraintes techniques dans le POC et testées.

ID	Contraintes	OK / KO
C1	Retourne en temps réel les coordonnées de géo localisation et les consignes de l'hôpital le plus proche en moins de 200ms.	OK
C2	Réutilisabilité du code et des techniques	OK
C3	Découpage en micro service permettant de tester individuellement chaque service de l'architecture.	OK
C4	API Restful	OK
C5	Intégration et déploiement continu en moins de 5 minutes	CI OK En 3 min

Feedback et Recommandations

Notes en guise de retour d'expérience

	Domaine	Note
1	Stack technique	Eclipse Entreprise Java 2021 + Spring Tool Suite STS Java JDK 17, Maven, Spring Boot, Spring MCV, Spring JPA (Hibernate), Lombok, JUnit, Jacoco, Surfire PostgreSQL, H2 in memory Jira, GitHub, GitHub Actions Postman, SonarCloud
2	Compétences techniques	Développeuse débutante sur l'architecture et le stack technique cible. Auto-formation avec cours OpenClassRooms Pas suffisamment de temps pour la formation, ni de support adapté. Pas de support technique pour accompagner en renfort. Pas de pair programming, ni de code review avec tiers.
3	Installation	Problèmes rencontrés lors de l'utilisation d'outils de développement ayant nécessité leurs réinstallation suite à de nombreux plantages: Eclipse, STS et PgAdmin.
4	PgAdmin4	Lenteurs sur l'utilisation, création et modification du schéma de base de données laborieux via l'interface. Problèmes rencontrés après avoir modifié le mot de passe de l'utilisateur par défaut postgres.
5	PostgreSQL	Problemes quand utilisation des majuscules pour nommer des tables ou des colonnes.
6	Jeux de données	La mise en place de jeux de données pertinents et adaptés aux tests a nécessité un temps d'effort relativement important qui a été sous-estimé. Créer la liste des pathologies, une liste d'hôpitaux, associer les pathologies aux hôpitaux et affecter nombre de lits en fonction des cas de test. Récupérer des adresses, avec la géo localisation (latitude, longitude) pour les hôpitaux et les patients. Volumétrie restreinte à 1 région 12 hôpitaux.
7	GitLab	Abandon des tests avec GitLab pour configurer le pipeline CI/CD (anomalies)
8	GitHub	Le repository GitHub a dû être recrée suite à des erreurs lors de l'intégration continue (ne trouvait pas le fichier Pom.xml qui n'était pas à la racine du repo)
9	Environnement Test et Recette	Pas d'architecture technique cible défini dans le DDA. Pas d'environnement de tests ni de recette mis à disposition pour réaliser les tests de performance, de montée en charge, de stress et de sécurité. Pas d'interface utilisateur permettant de tester notre API de bout en bout. Difficulté pour finaliser le déploiement continue sans environnement de Recette.

Recommandations

Les fonctionnalités pour l'API ERS

Ajouter un service permettant de récupérer les données de géo localisation du lieu d'intervention en fonction de son adresse, afin d'obtenir la latitude et la longitude.

<https://adresse.data.gouv.fr/api-doc/adresse>

https://developer.here.com/documentation/geocoding-search-api/dev_guide/index.html

<https://developers.google.com/maps/documentation/geocoding?hl=fr>

Ajouter une API permettant de calculer la distance entre l'adresse d'origine et la destination afin d'être plus précis sur le calcul de distance une fois la liste des 5 hôpitaux les plus proches aura été obtenue via la fonction de calcul de distance selon la formule d'Haversine.

<https://developers.google.com/maps/documentation/distance-matrix?hl=fr>

Amélioration : en cas d'alerte lorsqu'aucun hôpital proche de la zone d'intervention (région) n'a de lits disponibles alors effectuer automatiquement une nouvelle demande pour une région limitrophe.

Amélioration : proposer une interface utilisateur visuelle affichant sur une carte le lieu d'intervention et les hôpitaux aux alentours avec un code graphique permettant d'identifier rapidement les hôpitaux ayant des lits dans la spécialité demandée.

Plateforme Cloud Privé

Compléter l'architecture technique cible afin d'établir les technologies qui seront utilisées pour l'hébergement de l'application distribuée. Tel que défini dans le référentiel

[BETA - NHS digital, data and technology standards framework - NHS Digital](#)

Il est d'une importance vitale que les prestataires de santé et de soins aient accès à une connectivité de réseau de données hautement fiable, au meilleur rapport qualité-prix et de taille appropriée, capable de répondre à la demande croissante de services numériques. Une connectivité de réseau de données hautement performante, sécurisée et offrant le meilleur rapport qualité-prix dans le domaine de la santé et des soins. Avec des fournisseurs accrédités aux normes HSCN pour l'hébergement dans un Cloud privé.

Elle permettra de gérer les charges de travail de données avec la mise à l'échelle automatique, la haute disponibilité et la prise en charge de la reprise après sinistre. On pourra également mettre en place un plan de reprise qui devra être testé.

Les environnements de test:

Mettre à disposition des environnements complets pour les tests et la recette.

Cela permettra d'effectuer les tests de performance, de montée en charge et de stress et de sécurité.

De vérifier les KPI suivants :

- Disponibilité du système avec SLI $\geq 99,95\%$
- RPS (Request Per Second) : RPS ≥ 800 et 200ms/R
- Déploiement Continu (CD) de l'API dans l'environnement cible.
- Examens périodiques de la santé et des risques du système

Sécurité et confidentialité des données

Les API seront sécurisé derrière une API Gateway.

La connexion aux systèmes MedHead doit se faire via un système d'authentification approuvé. Les systèmes utilisés par le personnel du MedHead doivent vérifier que le personnel est authentifié et autorisé à utiliser les services de la plateforme.

Les API devront respecter le standard de données médicales FHIR (**Fast Healthcare Interoperability Resources**).

Mise en place systématique de DTO (Data Transfer Object) pour supprimer la vulnérabilité liée à l'exposition via les Endpoints de l'implémentation des entités.

Equipe / Compétences

Mettre en place des équipes Agiles avec des compétences seniors sur les stacks techniques (Spring Boot, Spring Security, JPA) en pair programming avec au moins 1 profil senior.

Outils de développement et de gestion de projet

Utiliser la plate-forme GitHub basée sur le cloud pour commenter toutes les modifications et les difficultés rencontrées et pouvoir récupérer tous ces commentaires. De cette façon, il est plus facile pour l'équipe du projet de participer et de collaborer. Utiliser l'outil de gestion de projet Jira pour noter les critères de réussite ou les mesures de gestion de projet clairement définis, les mesures d'évaluation, les échéanciers, les

prochains plans de gestion de projet (s'ils sont approuvés), les ressources nécessaires et d'autres aspects discutés. On y définira également les bugs.

KPI Post/Prod

Pour le KPI Taux de réussite : Plus de 90 % des cas d'urgence sont acheminés vers l'hôpital compétent le plus proche du réseau. On pourra correctement tester ce KPI une fois ERS en production, en analysant l'historique du journal des interventions.

Pour le KPI Fluidité du workflow : Le temps moyen de traitement d'une urgence passe de 18,25 minutes (valeur actuelle) à 12,00 minutes (valeur souhaitée) . On pourra correctement tester ce KPI une fois l'ensemble du traitement d'une urgence sera mis en place dont l'interface utilisateur, permettant de tester les fonctionnalités de bout en bout.

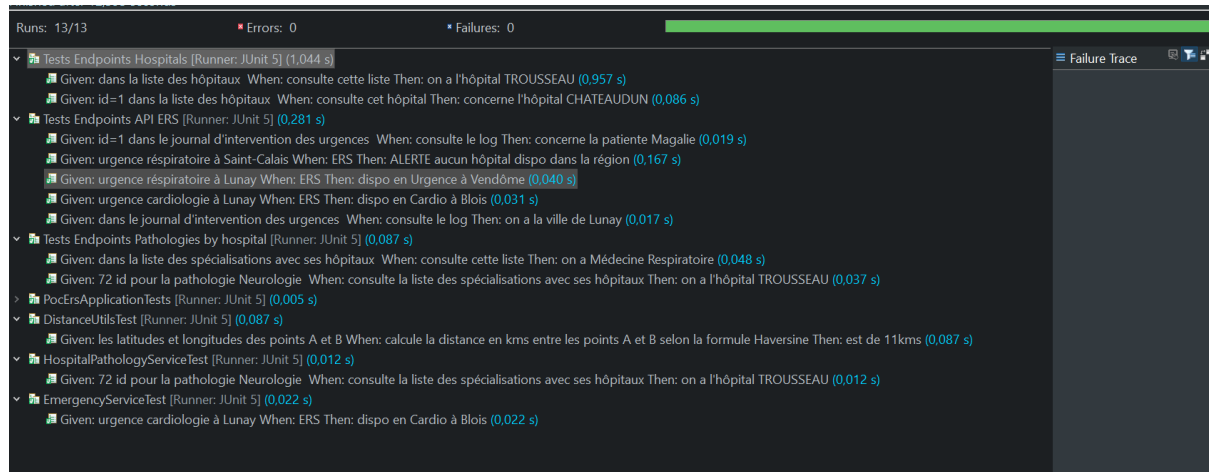
Pour le KPI Satisfaction des utilisateurs : La satisfaction des utilisateurs permet de mesurer la performance perçue par rapport à celles qu'ils attendaient. Note sur 5. On pourra correctement tester ce KPI une fois ERS en production.

Annexe

Résultats des tests réussis

Avec Postman JUnit :

Résultats des tests unitaires et d'intégration des scénarios des cas d'utilisation des fonctionnalités.



Avec Postman :

Résultat des tests avec la collection « POC API Emergency » permettant de tester les différents appels à l'API via des requêtes http (Get, Post).

POC API Emergency		No Environment, just now
RUN SUMMARY		
		1
▼ GET	http://localhost:9001/emergencies	2 0
	Pass Status code is 200	
	Pass Content-Type header is application/json	
▼ GET	http://localhost:9001/emergencies/1	3 0
	Pass Status code is 200	
	Pass Content-Type header is application/json	
	Pass Found patient Magalie when Emmergency log id = 1	
▼ POST	http://localhost:9001/emergencies	4 0
	Pass Status code is 200	
	Pass Content-Type header is application/json	
	Pass Hospital found in Blois	
	Pass Response time is less than 200ms	
▼ GET	http://localhost:9001/pathologies	2 0
	Pass Status code is 200	
	Pass Content-Type header is application/json	
▼ GET	http://localhost:9001/pathologies/70	2 0
	Pass Status code is 200	
	Pass Content-Type header is application/json	
▼ GET	http://localhost:9001/hospitals	2 0
	Pass Status code is 200	
	Pass Content-Type header is application/json	
▼ GET	http://localhost:9001/hospitals/1	3 0
	Pass Status code is 200	
	Pass Content-Type header is application/json	
	Pass Found hospital CHATEAUDUN when hospital id = 1	

Résultat des tests avec la collection « POC API Emergency Nearest Hospitals » permettant de tester les différents cas de test d'appel à l'API via la requêtes http (Post /emergencies) pour demander une intervention médicale urgente. Retourne en temps réel les coordonnées de géo localisation et les consignes de l'hôpital le plus proche.

POC API Emergency Nearest H... No Environment, just now

RUN SUMMARY

		1
▼ POST	ERS Lunay Cardio Blois	4 0
Pass Status code is 200		
Pass Content-Type header is application/json		
Pass Hospital found in Blois		
Pass Response time is less than 200ms		
▼ POST	ERS Lunay Respi Vendôme	5 0
Pass Status code is 200		
Pass Content-Type header is application/json		
Pass Hospital found in Vendôme		
Pass Service name is Urgence		
Pass Response time is less than 200ms		
▼ POST	ERS Respi Alerte pas de dispo	4 0
Pass Status code is 200		
Pass Hospital not found		
Pass Emmergency instructions Alerte found		
Pass Response time is less than 200ms		
▼ POST	ERS Monnaie Neuro Tours	4 0
Pass Status code is 200		
Pass Content-Type header is application/json		
Pass Hospital found in Tours		
Pass Response time is less than 200ms		
▼ POST	ERS Monnaie Pédiatrie Tours	4 0
Pass Status code is 200		
Pass Content-Type header is application/json		
Pass Hospital found in Tours		
Pass Response time is less than 200ms		
▼ POST	ERS Pithiviers Trauma Saran	4 0
Pass Status code is 200		
Pass Content-Type header is application/json		
Pass Hospital found in Saran		
Pass Response time is less than 200ms		

Résultat des tests avec la collection « POC API Emergency Loading Test » permettant de tester la montée en charge (sur un poste de développement). On exécute 800 requêtes (Post/Emergencies) en 66 secondes. Chaque requête s'exécute en moins de 20 ms.

POC API Emergency Stress Test				No Environment, a min ago
All Tests	Passed (3204)	Failed (0)		
Pass	Response time is less than 200ms			1
POST	ERS Lunay Pédiatrie 3	http://localhost:9001/emergencies / ERS Lunay Pédiatrie 3	200 OK 7 ms 951 B	2
Pass	Status code is 200			3
Pass	Hospital found			6
Pass	Hospital distance is less than 100kms			7
Pass	Response time is less than 200ms			8
POST	ERS Lunay Trauma 3	http://localhost:9001/emergencies / ERS Lunay Trauma 3	200 OK 10 ms 951 B	9
Pass	Status code is 200			10
Pass	Hospital found			11
Pass	Hospital distance is less than 100kms			12
Pass	Response time is less than 200ms			13
POST	ERS Respi Alerte 3	http://localhost:9001/emergencies / ERS Respi Alerte 3	200 OK 6 ms 846 B	14
Pass	Status code is 200			15
Pass	Hospital not found			17
Pass	Emergency instructions Alerte found			18
Pass	Response time is less than 200ms			19
POST	ERS Lunay Cardio 4	http://localhost:9001/emergencies / ERS Lunay Cardio 4	200 OK 8 ms 951 B	20
Pass	Status code is 200			21
Pass	Hospital found			22
Pass	Hospital distance is less than 100kms			23
Pass	Response time is less than 200ms			24
POST	ERS Lunay Neuro 4	http://localhost:9001/emergencies / ERS Lunay Neuro 4	200 OK 7 ms 966 B	25
Pass	Status code is 200			26
Pass	Hospital found			27
Pass	Hospital distance is less than 100kms			28
Pass	Response time is less than 200ms			29

Avec GitHub Actions :

Résultats de l'intégration continue sous GitHub Actions.

Déclenché automatiquement sur les Pull Request et les Push.


Update maven.yml #10

 Open Mamak2020 wants to merge 1 commit into `main` from `Pipeline-CICD` 



 Conversation 0  Commits 1  Checks 0  Files changed 1



Mamak2020 commented 4 minutes ago

Owner  ...

No description provided.

  Update maven.yml

Verified  8e6babb





sonarcloud bot commented 2 minutes ago

 ...


Kudos, SonarCloud Quality Gate passed! **Passed**

  0 Bugs
  0 Vulnerabilities
  0 Security Hotspots
  0 Code Smells

 No Coverage information
 No Duplication information

Add more commits by pushing to the `Pipeline-CICD` branch on Mamak2020/poc-api-ers.




 All checks have passed

[Hide all checks](#)

5 successful checks

-   CI/CD Pipeline / Build (pull_request) Successful in 15s [Details](#)
-   CI/CD Pipeline / Tests (pull_request) Successful in 26s [Details](#)
-   CI/CD Pipeline / SonarCloud analysis (pull_request) Successful in 59s [Details](#)
-   CI/CD Pipeline / Deploy (pull_request) Successful in 20s [Details](#)
-   SonarCloud Code Analysis Successful in 21s — Quality Gate passed [Details](#)

 This branch has no conflicts with the base branch

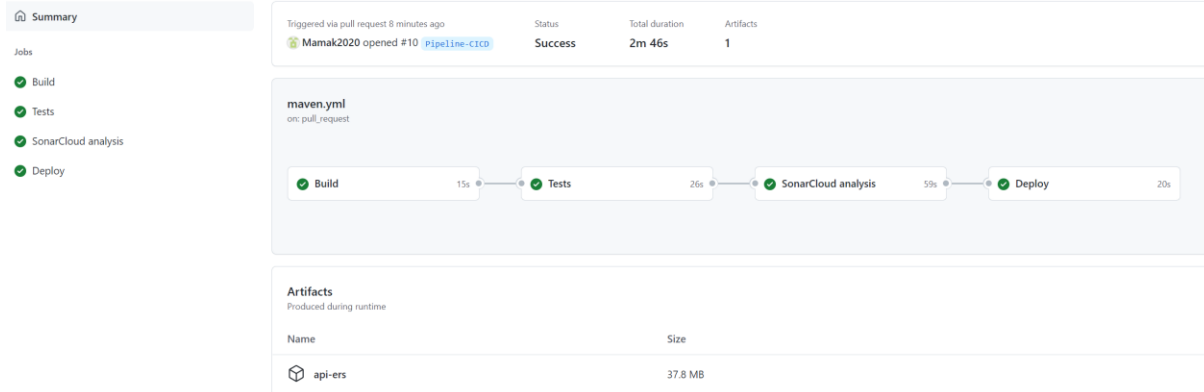
Merging can be performed automatically.

Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

L'ensemble du pipeline CI/CD s'exécute en moins de 3 minutes.

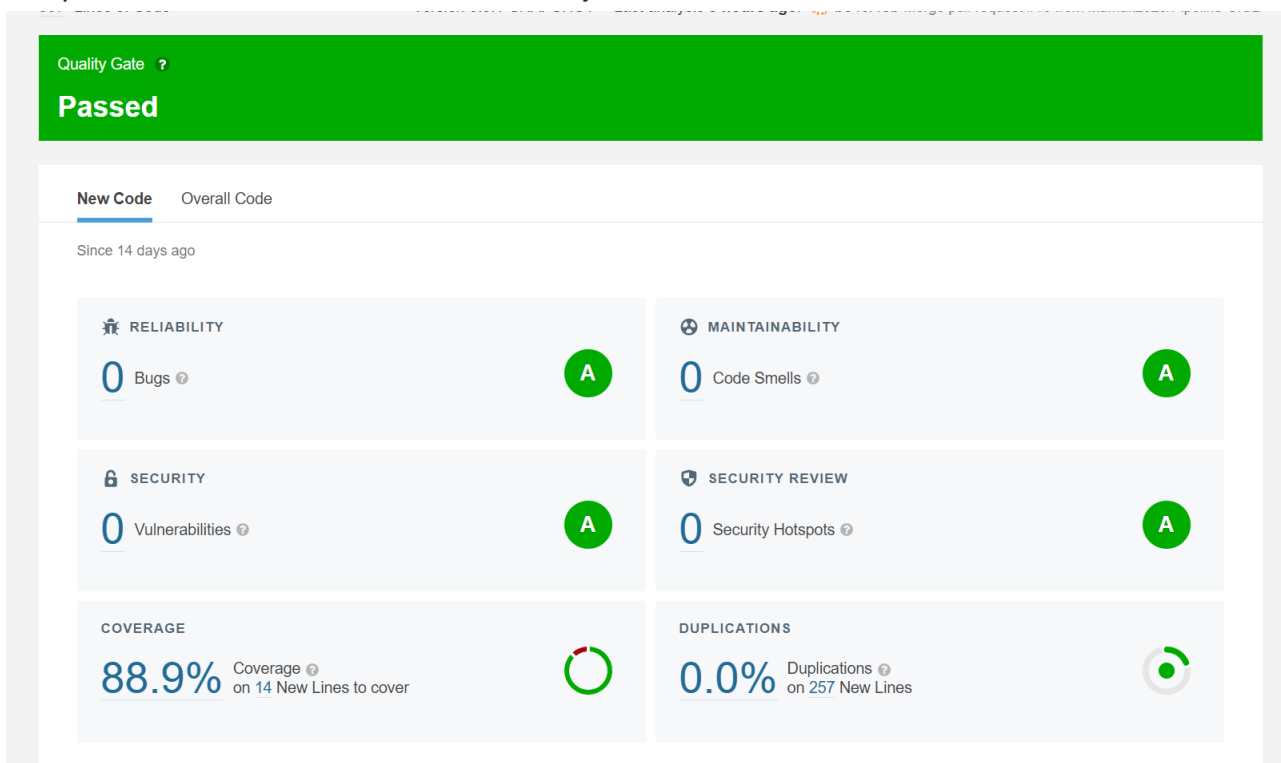
✓ Update maven.yml CI/CD Pipeline #66



Avec SonarCloud :

Résultats des tests de qualité avec SonarCloud.

La qualité du code de la dernière mise à jour est excellente.



La qualité du code dans son ensemble est excellente.



Remarque on a une vulnérabilité qu'il faudra corriger en remplaçant l'entité Emergency par un DTO, comme cela a été fait pour HospitalPathology

Project Overview

- Reliability
- Security **SECURITY RATING**
 - Overview
 - On new code
 - Vulnerabilities 0
 - Rating A
 - Remediation Effort 0
 - Overall
 - Vulnerabilities 1
 - Rating D
 - Remediation Effort 10min
- Security Review
- Maintainability
- Coverage
- Duplications
- Size
- Complexity
- Issues

ers / src / main / ... / com / medhead / ers / controller / EmergencyController.java

```
10 import org.springframework.web.bind.annotation.RestController;
11
12 import com.medhead.ers.model.Emergency;
13 import com.medhead.ers.service.EmergencyService;
14
15 @RestController
16 public class EmergencyController {
17     @Autowired
18     private EmergencyService emergencyService;
19
20     @GetMapping("/emergencies/{id}")
21     public Optional<Emergency> getEmergency(@PathVariable("id") final Long id) {
22         return emergencyService.getEmergency(id);
23     }
24
25     @GetMapping("/emergencies")
26     public Iterable<Emergency> getEmergencies() {
27         return emergencyService.getEmergencies();
28     }
29
30     @PostMapping("/emergencies")
31     public Emergency createEmergencyLog(@RequestBody Emergency emergency) {
32
33         return emergencyService.requestMedicalEmergency(emergency);
34     }
35 }
```

Replace this persistent entity with a simple POJO or DTO object. Why is this an issue? 13 days ago L31

Vulnerability Critical Open Magalie Morneau 10min effort No tags

La couverture des tests est de 100% sur nos principaux composants de notre API.

ers

View as

List

↑ ↓ to select files

← → to navigate

15 files

Coverage 24.3%

New code: since previous version

	Coverage	Uncovered Lines	Uncovered Conditions
src/main/java/com/medhead/ers/model/Pathology.java	2.3%	4	38
src/main/java/com/medhead/ers/model/Patient.java	2.7%	4	32
src/main/java/com/medhead/ers/model/Hospital.java	11.4%	0	62
src/main/java/com/medhead/ers/model/Emergency.java	11.5%	0	192
src/main/java/com/medhead/ers/model/HospitalPathology.java	11.6%	0	38
src/main/java/com/medhead/ers/service/PatientService.java	16.7%	2	13
src/main/java/com/medhead/ers/service/HospitalService.java	27.8%	0	13
src/main/java/com/medhead/ers/controller/PatientController.java	33.3%	2	–
src/main/java/com/medhead/ers/PocErsApplication.java	33.3%	2	–
src/main/java/com/medhead/ers/service/HospitalPathologyService.java	38.1%	0	13
src/main/java/com/medhead/ers/service/EmergencyService.java	68.0%	2	29
src/main/java/com/medhead/ers/utills/DistanceUtils.java	100%	0	–
src/main/java/com/medhead/ers/controller/EmergencyController.java	100%	0	–
src/main/java/com/medhead/ers/controller/HospitalController.java	100%	0	–
src/main/java/com/medhead/ers/controller/HospitalPathologyController.java	100%	0	–

15 of 15 shown